

Proceedings of the 25th International Workshop on Description Logics

June 7–10, 2012
Rome, Italy

EDITED BY
YEVGENY KAZAKOV, DOMENICO LEMBO, AND FRANK WOLTER

Preface

Welcome to the 25th International Workshop on Description Logics, DL 2012, in Rome, Italy. The workshop continues the long-standing tradition of international workshops devoted to discussing developments and applications of knowledge representation formalisms and systems based on Description Logics. The list of the International Workshops on Description Logics can be found at <http://dl.kr.org>. There were 58 papers submitted each of which was reviewed by at least three members of the program committee or additional reviewers recruited by the PC members. A total of 55 papers were selected for oral or poster presentation. The best student paper award was given to the paper *Elimination of Complex RIAs without Automata* by Frantisek Simancik from Oxford University. In addition to the presentation of the accepted papers, posters, and demos the following invited talks were given at the workshop:

- Serge Abiteboul (Collège de France, INRIA Saclay & ENS Cachan),
Viewing the Web as a Distributed Knowledge Base.
- Piero Bonatti (Università degli Studi di Napoli “Federico II”, Italy),
Defaults in Description Logics: So Simple, So Difficult.
- Alan Rector (University of Manchester, U.K.),
What’s missing? DLs, OWL and the Ecology of Semantic Systems.

The talk by Piero Bonatti was given as part of a joint session with the 14th International Workshop on Non-Monotonic Reasoning. Our thanks go to all the authors for submitting to DL, and to the invited speakers, PC members, and all additional reviewers who made the technical programme possible. DL 2012 was sponsored by the Artificial Intelligence Journal and by Principles of Knowledge Representation and Reasoning, Incorporated (KR). We would like to acknowledge that the work of the PC was greatly simplified by using the EasyChair conference management system (www.easychair.org) developed by Andrei Voronkov.

Yevgeny Kazakov, Domenico Lembo, and Frank Wolter
DL 2012 Chairs

Organizing Committee

General Chair

Domenico Lembo University of Rome “La Sapienza”

Programm Chairs

Yevgeny Kazakov University of Ulm
Frank Wolter University of Liverpool

Program Committee

Carlos Areces	FaMAF - Universidad Nacional de Córdoba
Alessandro Artale	Free University of Bozen-Bolzano
Franz Baader	TU Dresden
Meghyn Bienvenu	CNRS & Université Paris-Sud
Alex Borgida	Rutgers University
Diego Calvanese	Free University of Bozen-Bolzano
Bernardo Cuenca Grau	University of Oxford
Giuseppe De Giacomo	University of Rome “La Sapienza”
Enrico Franconi	Free University of Bozen-Bolzano
Birte Glimm	University of Ulm
Valentin Goranko	Technical University of Denmark
Rajeev Gore	The Australian National University
Ian Horrocks	University of Oxford
Yevgeny Kazakov	University of Ulm
Pavel Klinov	University of Arizona & Clark and Parsia, LLC
Boris Konev	University of Liverpool
Roman Kontchakov	Birkbeck College
Markus Krötzsch	University of Oxford
Maurizio Lenzerini	University of Rome “La Sapienza”
Thorsten Liebig	derivo GmbH
Carsten Lutz	University of Bremen
Maarten Marx	University of Amsterdam
Thomas Meyer	Meraka Institute
Boris Motik	University of Oxford
Ralf Möller	Hamburg University of Technology
Magdalena Ortiz	Vienna University of Technology
Bijan Parsia	University of Manchester
Peter Patel-Schneider	Bell Labs Research
Rafael Peñaloza	TU Dresden
Riccardo Rosati	University of Rome “La Sapienza”

Sebastian Rudolph	Karlsruhe Institute of Technology
Ulrike Sattler	University of Manchester
Renate A. Schmidt	University of Manchester
Thomas Schneider	University of Bremen
Luciano Serafini	FBK-IRST
Mantas Simkus	Vienna University of Technology
Viorica Sofronie-Stokkermans	University Koblenz-Landau
Giorgos Stamou	National Technical University of Athens
Giorgos Stoilos	University of Oxford
Umberto Straccia	ISTI-CNR
David Toman	University of Waterloo
Anni-Yasmin Turhan	TU Dresden
Grant Weddell	University of Waterloo
Frank Wolter	University of Liverpool
Michael Zakharyashev	Birkbeck College London

Additional Reviewers

Ana Armas	Jens Lehmann
Stefan Borgwardt	Despoina Magka
Loris Bozzato	Kody Moodley
Arina Britz	Linh Anh Nguyen
Giovanni Casini	Özgür Lütfü Özcep
Alexandros Chortaras	Ian Pratt-Hartmann
Aurona Gerber	Mariano Rodriguez-Muro
Silvio Ghilardi	Patrik Schneider
Víctor Didier Gutiérrez Basulto	Inanc Seylan
Matthew Horridge	Frantisek Simancik
Yazmin Angelica Ibanez-Garcia	Giorgos Stamou
Jean Christoph Jung	Dmitry Tsarkov
Mohammad Khodadadi	Zhe Wang
Ilianna Kollia	Yujiao Zhou
Clemens Kupke	

Table of Contents

Invited Talks

Viewing the Web as a Distributed Knowledge Base	1
<i>Serge Abiteboul</i>	
Defaults in Description Logics: So Simple, So Difficult	2
<i>Piero Bonatti</i>	
What’s missing? DLs, OWL and the Ecology of Semantic Systems	3
<i>Alan Rector</i>	

Full Papers

Representability in $DL-Lite_{\mathcal{R}}$ Knowledge Base Exchange	4
<i>Marcelo Arenas, Elena Botoeva, Diego Calvanese, Vladislav Ryzhikov and Evgeny Sherkhonov</i>	
Modular Combination of Reasoners for Ontology Classification	15
<i>Ana Armas Romero, Bernardo Cuenca Grau and Ian Horrocks</i>	
UEL: Unification Solver for \mathcal{EL}	26
<i>Franz Baader, Stefan Borgwardt, Julian Mendez and Barbara Morawska</i>	
A Goal-Oriented Algorithm for Unification in \mathcal{EL} w.r.t. Cycle-Restricted TBoxes	37
<i>Franz Baader, Stefan Borgwardt and Barbara Morawska</i>	
Diversity of Reason: Equivalence Relations over Description Logic Explanations	48
<i>Samantha Bail, Bijan Parsia and Ulrike Sattler</i>	
Inconsistency-Tolerant Conjunctive Query Answering for Simple Ontologies	59
<i>Meghyn Bienvenu</i>	
Deciding FO-Rewritability in \mathcal{EL}	70
<i>Meghyn Bienvenu, Carsten Lutz and Frank Wolter</i>	
Answering Expressive Path Queries over Lightweight DL Knowledge Bases	81
<i>Meghyn Bienvenu, Magdalena Ortiz and Mantas Simkus</i>	
Experiences in Mapping the Business Intelligence Model to Description Logics, and the Case for Parametric Concepts	92
<i>Alexander Borgida, Jennifer Horkoff, John Mylopoulos and Riccardo Rosati</i>	
Gödel Negation Makes Unwitnessed Consistency Crisp	103
<i>Stefan Borgwardt, Felix Distel and Rafael Peñaloza</i>	

Towards More Effective Tableaux Reasoning for CKR	114
<i>Loris Bozzato, Martin Homola and Luciano Serafini</i>	
Inverting Subsumption for Constructive Reasoning	125
<i>Simona Colucci and Francesco M. Donini</i>	
An ExpSpace Tableau-based Algorithm for <i>SHOIQ</i>	136
<i>Chan Le Duc, Myriam Lamolle and Olivier Cure</i>	
Role-depth Bounded Least Common Subsumers for \mathcal{EL}^+ and \mathcal{ELI}	147
<i>Andreas Ecke and Anni-Yasmin Turhan</i>	
Towards Practical Query Answering for Horn- <i>SHIQ</i>	158
<i>Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran and Guohui Xiao</i>	
Exact Query Reformulation over <i>SHOQ</i> DBoxes	169
<i>Enrico Franconi, Volha Kerhet and Nhung Ngo</i>	
Preferential Low Complexity Description Logics: Complexity Results and Proof Methods	180
<i>Laura Giordano, Valentina Gliozzi, Nicola Olivetti and Gian Luca Pozzato</i>	
Concept-Based Semantic Difference in Expressive Description Logics	191
<i>Rafael S. Gonçalves, Bijan Parsia and Ulrike Sattler</i>	
Equality-Friendly Well-Founded Semantics and Applications to Description Logics	202
<i>Georg Gottlob, André Hernich, Clemens Kupke and Thomas Lukasiewicz</i>	
Finite Model Reasoning in DL-Lite with Cardinality Constraints	213
<i>Yazmin Angelica Ibanez-Garcia</i>	
An Abstract Tableau Calculus for the Description Logic <i>SHOI</i> Using Unrestricted Blocking and Rewriting	224
<i>Mohammad Khodadadi, Renate A. Schmidt and Dmitry Tishkovsky</i>	
Long Rewritings, Short Rewritings	235
<i>Stanislav Kikot, Roman Kontchakov, Vladimir Podolskii and Michael Zakharyashev</i>	
Cost Based Query Ordering over OWL Ontologies	246
<i>Ilianna Kollia and Birte Glimm</i>	
Inconsistency-Tolerant First-Order Rewritability of DL-Lite with Identification and Denial Assertions	257
<i>Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi and Domenico Fabio Savo</i>	

Mixing Open and Closed World Assumption in Ontology-Based Data Access: Non-Uniform Data Complexity	268
<i>Carsten Lutz, Inanc Seylan and Frank Wolter</i>	
Modelling Structured Domains Using Description Graphs and Logic Programming	279
<i>Despoina Magka, Boris Motik and Ian Horrocks</i>	
Axiom Pinpointing Using an Assumption-Based Truth Maintenance System	290
<i>Hai H. Nguyen, Natasha Alechina and Brian Logan</i>	
Combining DL-Lite with Spatial Calculi for Feasible Geo-thematic Query Answering	301
<i>Özgür Lütü Özcep and Ralf Möller</i>	
OCL-Lite: A Decidable (Yet Expressive) Fragment of OCL	312
<i>Anna Queralt, Alessandro Artale, Diego Calvanese and Ernest Teniente</i>	
Query Rewriting under Extensional Constraints in DL-Lite	323
<i>Riccardo Rosati</i>	
Elimination of Complex RIAs without Automata	334
<i>Frantisek Simancik</i>	
Improved Algorithms for Module Extraction and Atomic Decomposition .	345
<i>Dmitry Tsarkov</i>	
Incremental Query Rewriting for OWL 2 QL	356
<i>Tassos Venetis, Giorgos Stoilos and Giorgos Stamou</i>	
Absorption for ABoxes	367
<i>Jiewen Wu, Alexander Hudek, David Toman and Grant Weddell</i>	
A Parallel Reasoner for the Description Logic \mathcal{ALC}	378
<i>Kejia Wu and Volker Haarslev</i>	
Towards an Expressive Decidable Logical Action Theory	389
<i>Wael Yehia and Mikhail Soutchanski</i>	
Poster Papers	
Concurrent Classification of OWL Ontologies - An Empirical Evaluation .	400
<i>Mina Aslani and Volker Haarslev</i>	
Non-Gödel Negation Makes Unwitnessed Consistency Undecidable	411
<i>Stefan Borgwardt and Rafael Peñaloza</i>	

ORM2 Encoding into Description Logic (Extended Abstract)	422
<i>Enrico Franconi, Alessandro Mosca and Dmitry Solomakhin</i>	
Adding Context to Tableaux for DLs	432
<i>Weili Fu and Rafael Peñaloza</i>	
Naïve ABox abduction in \mathcal{ALC} using a DL tableau	443
<i>Ken Halland and Katarina Britz</i>	
Patent Valuation Using Difference in \mathcal{ALEN}	454
<i>Naouel Karam and Adrian Paschke</i>	
A Formal Characterization of Concept Learning in Description Logics . . .	464
<i>Francesca A. Lisi</i>	
Nonmonotonic Reasoning in Description Logic by Tableaux Algorithm with Blocking	475
<i>Jaromir Malenko and Petr Stepanek</i>	
A Protégé Plug-in for Defeasible Reasoning	486
<i>Kody Moodley, Thomas Meyer and Ivan Varzinczak</i>	
A Decidable Extension of \mathcal{SRIQ} with Disjunctions in Complex Role Inclusion Axioms	497
<i>Milenko Mosurovic, Henson Graves and Nenad Krdzavac</i>	
Optimising Parallel ABox Reasoning of \mathcal{EL} Ontologies	508
<i>Yuan Ren, Jeff Z. Pan and Kevin Lee</i>	
Probabilistic Datalog+/- under the Distribution Semantics	519
<i>Fabrizio Riguzzi, Elena Bellodi and Evelina Lamma</i>	
Algebraic Reasoning for \mathcal{SHIQ}	530
<i>Laleh Roosta Pour and Volker Haarslev</i>	
Small Datalog Query Rewritings for \mathcal{EL}	541
<i>Giorgio Stefanoni, Boris Motik and Ian Horrocks</i>	
Extended Caching and Backjumping for Expressive Description Logics . . .	552
<i>Andreas Steigmiller, Thorsten Liebig and Birte Glimm</i>	
From \mathcal{EL} to Tractable Existential Rules with Complex Role Inclusions . . .	563
<i>Michaël Thomazo</i>	
Logical Relevance in Ontologies	574
<i>Chiara Del Vescovo, Bijan Parsia and Ulrike Sattler</i>	

Experimental Results on Solving the Projection Problem in Action Formalisms Based on Description Logics	585
<i>Wael Yehia, Hongkai Liu, Marcel Lippmann, Franz Baader and Mikhail Soutchanski</i>	
Efficient Upper Bound Computation of Query Answers in Expressive Description Logics	596
<i>Yujiao Zhou, Bernardo Cuenca Grau and Ian Horrocks</i>	

Viewing the Web as a Distributed Knowledge Base

Serge Abiteboul

Collège de France, INRIA Saclay & ENS Cachan, France

Information of interest may be found on the Web in a variety of forms, in many systems, and with different access protocols. A typical user may have information on many devices (smartphone, laptop, TV box, etc.), many systems (mailers, blogs, Web sites, etc.), many social networks (Facebook, Picasa, etc.). This same user may have access to more information from family, friends, associations, companies, and organizations. Today, the control and management of the diversity of data and tasks in this setting are beyond the skills of casual users. Facing similar issues, companies see the cost of managing and integrating information skyrocketing.

We are interested here in the management of such data. Our focus is not on harvesting all the data of a particular user or a group of users and then managing it in a centralized manner. Instead, we are concerned with the management of Web data in place in a distributed manner, with a possibly large number of autonomous, heterogeneous systems collaborating to support certain tasks.

Our thesis is that managing the richness and diversity of user-centric data residing on the Web can be tamed using a holistic approach based on a distributed knowledge base. All Web informations are represented as logical facts, and Web data management tasks as logical rules. We discuss Webdamlog, a variant of datalog for distributed data management that we use for this purpose. The automatic reasoning provided by its inference engine, operating over the Web knowledge base, greatly benefits a variety of complex data management tasks that currently require intense work and deep expertise.

Defaults in Description Logics: So Simple, So Difficult

Piero Bonatti

Università degli Studi di Napoli “Federico II”, Italy

Frame systems—the ancestors of Description Logics—supported a form of defeasible inheritance and overriding. Such nonmonotonic features disappeared from implementations after the logical reconstruction of frame systems, although applications provide interesting use cases for nonmonotonic inferences. This talk gives an overview of the nonmonotonic DLs introduced so far, and illustrates the many complexity issues that affect them (which probably explain the lack of support to nonmonotonic reasoning in DL reasoners). Research never surrenders, though: a pragmatic change of perspective yields encouraging results, that bring the quest for low-complexity, nonmonotonic DLs closer to its goal.

What's missing?

DLs, OWL and the Ecology of Semantic Systems

Alan Rector

University of Manchester, U.K.

Description logics are about to take off. Or are they? We've said it before. "Ontologies" and "OWL" have become buzz words. But there are barriers for anyone not in a centre of DL expertise, and sometimes even there. We use DLs/OWL in our commercial collaborations to manage concept composition, heterogeneity, indexing and context. We do not see how to do without them. In some areas, progress has been stunning. However, we still find gaps, e.g.: a) expressiveness and interaction with other knowledge representation paradigms b) Interaction with software engineering, c) tooling and user-friendly "intermediate representations" d) predictability and stability. This talk deals with the first three.

Before DLs emerged in the 1980s, most Knowledge Representation Systems were massively hybrid. They were messy, heuristic, certainly neither complete nor decidable. DLs have brought rigour but at the cost of a narrow focus, often too narrow we argue. Ontologies/DL models are not all of knowledge representation. Most knowledge is particular rather than universal; much is probabilistic, possibilistic, heuristic, or just navigational. Many other modelling paradigms - e.g. Frames, UML, RDF(S), Object oriented programming - are template based whereas DLs are axiom based. How do we bridge the gaps?

Most users and many software engineers naturally express and understand their knowledge at higher level of abstraction than raw DLs. How do we provide them with appropriate "intermediate representations"? How do we best build on their existing software engineering expertise? How do we be clear about when DLs are not suitable? In short, how do we embed DLs in an effective ecology of semantic systems?

Representability in $DL\text{-}Lite_{\mathcal{R}}$ Knowledge Base Exchange

M. Arenas¹, E. Botoeva², D. Calvanese², V. Ryzhikov², and E. Sherkhonov³

¹ Dept. of Computer Science, PUC Chile
marenas@ing.puc.cl

² KRDB Research Centre, Free Univ. of Bozen-Bolzano, Italy
lastname@inf.unibz.it

³ ISLA, University of Amsterdam, Netherlands
e.sherkhonov@uva.nl

Abstract. Knowledge base exchange can be considered as a generalization of data exchange in which the aim is to exchange between a source and a target connected through mappings, not only explicit knowledge, i.e., data, but also implicit knowledge in the form of axioms. Such problem has been investigated recently using Description Logics (DLs) as representation formalism, thus assuming that the source and target KBs are given as a DL TBox+ABox, while the mappings have the form of DL TBox assertions. In this paper we are interested in the problem of representing a given source TBox by means of a target TBox that captures at best the intensional information in the source. In previous work, results on representability have been obtained for $DL\text{-}Lite_{RDFS}$, a DL corresponding to the FOL fragment of RDFS. We extend these results to the positive fragment of $DL\text{-}Lite_{\mathcal{R}}$, in which, differently from $DL\text{-}Lite_{RDFS}$, the assertions in the TBox and the mappings may introduce existentially implied individuals. For this we need to overcome the challenge that the chase, a key notion in data and knowledge base exchange, is not guaranteed anymore to be finite.

1 Introduction

Knowledge base exchange is an extension of the data exchange setting, where the source may contain implicit knowledge by which new data may be inferred. The first framework for data exchange with incompletely specified data in the source was proposed in [3]. This framework is based on the general notion of *representation system*, as a mechanism to represent multiple instances of a data schema, and considers the problem of incomplete data exchanges under mappings constituted by a set of tuple generating dependencies (tgds), i.e., mappings between pairs of conjunctive queries. Given that the source data may be incompletely specified, (possibly infinitely) many source instances are implicitly represented. This framework was refined in [1, 2] to the case where as a representation system Description Logics (DL) knowledge bases (KBs) were used: the TBox and the ABox of a DL KB represent implicit and explicit information respectively, and mappings are sets of DL inclusions. While in the traditional data exchange setting, given a source instance and a mapping specification, (*universal*) *solutions* are target instances derived from the source instance and the mapping, in this case solutions are target DL KBs, derived from the source KB and the mapping.

In such a setting, in order to minimize the exchange (and hence transfer and materialization) of explicit (i.e., ABox) information, one is interested in computing universal solutions that contain as much implicit knowledge as possible. Therefore, the notion of *representability* was defined, which helps us in understanding the capacity of (universal) solutions to transfer implicit knowledge: we say that a source TBox is representable under a mapping if there exists a target TBox that leads to a universal solution when it is combined with a suitable ABox computed from the source ABox, independently of the actual source ABox. *Weak representability* is concerned with representability under a mapping extended with assertions that are implied by the given mapping and the source TBox. (Weak) representability of a source TBox under a mapping implies that the only knowledge that remains to be transferred explicitly via the (extended) mapping is the one in the source ABox. Therefore, checking (weak) representability and computing a representation of a source TBox under a(n extended) mapping turn out to be crucial problems in the context of KB exchange.

In [1, 2] the problems of deciding representability and weak-representability, and of computing a representation for a given mapping and a TBox was tackled for $DL-Lite_{RDFS}$, the DL counterpart of RDFS [5] and a member of the *DL-Lite* family of DLs [6]. It has been shown that these problems can be solved in polynomial time for $DL-Lite_{RDFS}$ mappings and TBoxes. Moreover, due to the simplicity of the logic, the characterization of representations is concise and simple.

In this paper we extend those results to the case of $DL-Lite_{\mathcal{R}}$ without disjointness assertions, a DL that we call $DL-Lite_{\mathcal{R}}^{pos}$. The presence of existential quantifiers on the right-hand side of concept inclusions makes the problem considerably more complicated than for $DL-Lite_{RDFS}$. However, we show that also in the presence of existentials on the right-hand side we are able to decide representability and weak representability of a $DL-Lite_{\mathcal{R}}^{pos}$ TBox under a $DL-Lite_{\mathcal{R}}^{pos}$ mapping in polynomial time and to construct a polynomial size representation.

2 Preliminaries

2.1 $DL-Lite_{\mathcal{R}}^{pos}$ Knowledge Bases

The DLs of the *DL-Lite* family [6] are characterized by the fact that reasoning can be done in polynomial time, and that data complexity of reasoning and conjunctive query answering is in AC^0 . We present now the syntax and semantics of $DL-Lite_{\mathcal{R}}^{pos}$, which is the DL that we adopt here, together with a sub-language of it.

In the following, we use A and P to denote concept and role names, respectively, and B and R to denote generic concepts and roles, respectively. The latter are defined by the following grammar:

$$R ::= P \mid P^- \qquad B ::= A \mid \exists R$$

For a role R , we use R^- to denote P^- when $R = P$, and P when $R = P^-$.

A $DL-Lite_{\mathcal{R}}^{pos}$ TBox is a finite set of concept inclusions $B \sqsubseteq B'$ and role inclusions $R \sqsubseteq R'$. A $DL-Lite_{\mathcal{R}}^{pos}$ ABox is a finite set of membership assertions of the form $A(u)$ and $P(u, v)$, where u and v are *individuals* or *labeled nulls*. We distinguish between the

two, since individuals are interpreted under the unique name assumption, while labeled nulls obtain their meaning through assignments (see below). Notice that we include labeled nulls in ABoxes as they are needed in the exchange of KBs. A $DL\text{-Lite}_{\mathcal{R}}^{pos}$ KB \mathcal{K} is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where \mathcal{T} is a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ TBox and \mathcal{A} is a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ ABox.

Note that $DL\text{-Lite}_{\mathcal{R}}^{pos}$ is the fragment of the DL $DL\text{-Lite}_{\mathcal{R}}$ studied in [6] without disjointness assertions on concepts and roles. In $DL\text{-Lite}_{\mathcal{R}}$, B and R are called *basic* concepts and *basic* roles, respectively, and for coherence with previous work on the $DL\text{-Lite}$ family, we adopt here this terminology as well. We call $DL\text{-Lite}_{\mathcal{R}DFS}$ the fragment of $DL\text{-Lite}_{\mathcal{R}}^{pos}$ (and hence of $DL\text{-Lite}_{\mathcal{R}}$) in which there are only atomic concepts and atomic roles on the right-hand side of inclusions.

The semantics of $DL\text{-Lite}_{\mathcal{R}}^{pos}$ is, as usual in DLs, based on the notion of first-order interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty domain and $\cdot^{\mathcal{I}}$ is an interpretation function such that: (1) $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, for every concept name A ; (2) $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, for every role name P ; (3) $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, for every individual name a ; and (4) such that:

$$\begin{aligned} (\exists R)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \text{there exists } y \in \Delta^{\mathcal{I}} \text{ s.t. } (x, y) \in R^{\mathcal{I}}\} \\ (P^-)^{\mathcal{I}} &= \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in P^{\mathcal{I}}\} \end{aligned}$$

Moreover, satisfaction of concept and role inclusions is defined as follows: $\mathcal{I} \models B \sqsubseteq B'$ if $B^{\mathcal{I}} \subseteq B'^{\mathcal{I}}$, and $\mathcal{I} \models R \sqsubseteq R'$ if $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$. Finally, satisfaction of membership assertions is defined as follows. A *substitution* over an interpretation \mathcal{I} is a function h from individuals and labeled nulls to $\Delta^{\mathcal{I}}$ such that $h(a) = a^{\mathcal{I}}$ for each individual a . Then $(\mathcal{I}, h) \models A(u)$ if $h(u) \in A^{\mathcal{I}}$, and $(\mathcal{I}, h) \models P(u, v)$ if $(h(u), h(v)) \in P^{\mathcal{I}}$.

An interpretation \mathcal{I} is a *model* of a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ TBox \mathcal{T} if for every $\alpha \in \mathcal{T}$, it holds that $\mathcal{I} \models \alpha$, and it is a *model* of a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ ABox \mathcal{A} if there exists a substitution h over \mathcal{I} such that for every $\alpha \in \mathcal{A}$, it holds that $(\mathcal{I}, h) \models \alpha$. Finally, \mathcal{I} is a *model* of a $DL\text{-Lite}_{\mathcal{R}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if \mathcal{I} is a model of both \mathcal{T} and \mathcal{A} . The set of all models of \mathcal{K} is denoted $\text{MOD}(\mathcal{K})$, and \mathcal{K} is *consistent* if $\text{MOD}(\mathcal{K}) \neq \emptyset$. We observe that in $DL\text{-Lite}_{\mathcal{R}}^{pos}$ one cannot express any form of negative information, and hence a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ KB is always consistent.

We assume that interpretations satisfy the standard names assumption, that is, we assume given a fixed infinite set \mathbf{U} of individual names, and we assume that for every interpretation \mathcal{I} , it holds that $\Delta^{\mathcal{I}} \subseteq \mathbf{U}$ and $a^{\mathcal{I}} = a$ for every individual name a . This implies that interpretations satisfy the unique name assumption over individual names.

A *signature* Σ is a set of concept and role names. An interpretation \mathcal{I} is said to be an interpretation of Σ if it is defined exactly on the concept and role names in Σ . Given a KB \mathcal{K} , the *signature* $\Sigma(\mathcal{K})$ of \mathcal{K} is the alphabet of concept and role names occurring in \mathcal{K} , and \mathcal{K} is said to be *defined over* (or simply, *over*) a signature Σ if $\Sigma(\mathcal{K}) \subseteq \Sigma$ (and likewise for a TBox \mathcal{T} , an ABox \mathcal{A} , inclusions $B \sqsubseteq C$ and $R \sqsubseteq Q$, and membership assertions $A(u)$ and $P(u, v)$).

2.2 Queries, Certain Answers, and Chase

A k -ary query q over a signature Σ , with $k \geq 0$, is a function that maps every interpretation $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of Σ into a k -ary relation $q^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}^k}$. In particular, if $k = 0$, then q is called a Boolean query, and $q^{\mathcal{I}}$ is either a relation containing the empty tuple $()$ (representing

the value true) or the empty relation (representing the value false). A query q is said to be a query over a KB \mathcal{K} if q is a query over a signature Σ and $\Sigma \subseteq \Sigma(\mathcal{K})$. Moreover, the answer to q over \mathcal{K} , denoted by $\text{cert}(q, \mathcal{K})$, is defined as $\text{cert}(q, \mathcal{K}) = \bigcap_{\mathcal{I} \in \text{MOD}(\mathcal{K})} q^{\mathcal{I}}$. Each tuple in $\text{cert}(q, \mathcal{K})$ is called a *certain answer* for q over \mathcal{K} . Notice that if q is a Boolean query, then $\text{cert}(q, \mathcal{K})$ evaluates to true if $q^{\mathcal{I}}$ evaluates to true for every $\mathcal{I} \in \text{MOD}(\mathcal{K})$, and it evaluates to false otherwise.

In this paper, we adopt the class of unions of conjunctive queries as our main query formalism. A *conjunctive query (CQ)* over a signature Σ is a first-order formula of the form $q(\mathbf{x}) = \exists \mathbf{y}. \text{conj}(\mathbf{x}, \mathbf{y})$, where \mathbf{x}, \mathbf{y} are tuples of variables and $\text{conj}(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms of the form: (1) $A(t)$, with A a concept name in Σ and t either an individual from \mathbf{U} or a variable from \mathbf{x} or \mathbf{y} , or (2) $P(t_1, t_2)$, with P a role name in Σ and t_i ($i = 1, 2$) either an individual from \mathbf{U} or a variable from \mathbf{x} or \mathbf{y} . In a CQ $q(\mathbf{x}) = \exists \mathbf{y}. \text{conj}(\mathbf{x}, \mathbf{y})$ over a signature Σ , \mathbf{x} is the tuple of free variables of $q(\mathbf{x})$. Moreover, given an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of Σ , the answer of q over \mathcal{I} , denoted by $q^{\mathcal{I}}$, is defined as the set of tuples \mathbf{a} of elements from $\Delta^{\mathcal{I}}$ for which there exist a tuple \mathbf{b} of elements from $\Delta^{\mathcal{I}}$ such that \mathcal{I} satisfies every conjunct in $\text{conj}(\mathbf{a}, \mathbf{b})$. Finally, a union of conjunctive queries (UCQ) over a signature Σ is a finite set of CQs over Σ that have the same free variables. A UCQ $q(\mathbf{x})$ is interpreted as the first-order formula $\bigvee_{q_i \in q} q_i(\mathbf{x})$, and its semantics is defined as $q^{\mathcal{I}} = \bigcup_{q_i \in q} q_i^{\mathcal{I}}$.

Certain answers in $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ can be characterized through the notion of chase. We call a *chase* a (possibly infinite) set of assertions of the form $A(t), P(t, t')$, where t, t' are either individuals from \mathbf{U} , or labeled nulls interpreted as not necessarily distinct domain elements (see the definition of the semantics of $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ in Section 2.1). For $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ KBs, we employ the notion of restricted chase as defined in [6]. For such a KB $\langle \mathcal{T}, \mathcal{A} \rangle$, the chase of \mathcal{A} w.r.t. \mathcal{T} , denoted $\text{chase}_{\mathcal{T}}(\mathcal{A})$, is a chase obtained from \mathcal{A} by adding facts implied by inclusions in \mathcal{T} , and introducing fresh labeled nulls whenever required by an inclusion with $\exists R$ in the right-hand side (see [6] for details).

2.3 Knowledge Base Exchange Framework

Assume that Σ_1, Σ_2 are signatures with no concepts or roles in common. Then we say that an inclusion $N_1 \sqsubseteq N_2$ is an *inclusion from Σ_1 to Σ_2* , if N_1 is a concept or a role over Σ_1 and N_2 is a concept or a role over Σ_2 . For a DL \mathcal{L} (e.g., $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$), we define an \mathcal{L} -*mapping* (or just *mapping*, when \mathcal{L} is clear from the context) as a tuple $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, where \mathcal{T}_{12} is a TBox in \mathcal{L} consisting of inclusions from Σ_1 to Σ_2 :

- (1) $C_1 \sqsubseteq C_2$, where C_1, C_2 are concepts in \mathcal{L} over Σ_1 and Σ_2 , respectively, and
- (2) $Q_1 \sqsubseteq Q_2$, where Q_1 and Q_2 are roles in \mathcal{L} over Σ_1 and Σ_2 , respectively.

If \mathcal{T}_{12} is an \mathcal{L} -TBox, for a DL \mathcal{L} (e.g., $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$), then \mathcal{M} is called an \mathcal{L} -*mapping*.

The semantics of a mapping is defined in terms of the notion of satisfaction. More specifically, given a mapping $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, an interpretation \mathcal{I} of Σ_1 and an interpretation \mathcal{J} of Σ_2 , pair $(\mathcal{I}, \mathcal{J})$ *satisfies* TBox \mathcal{T}_{12} , denoted by $(\mathcal{I}, \mathcal{J}) \models \mathcal{T}_{12}$, if for each concept inclusion $C_1 \sqsubseteq C_2 \in \mathcal{T}_{12}$, it holds that $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{J}}$, and for each role inclusion $Q_1 \sqsubseteq Q_2 \in \mathcal{T}_{12}$, it holds that $Q_1^{\mathcal{I}} \subseteq Q_2^{\mathcal{J}}$. Moreover, given an interpretation \mathcal{I} of Σ_1 , $\text{SAT}_{\mathcal{M}}(\mathcal{I})$ is defined as the set of interpretations \mathcal{J} of Σ_2 such

that $(\mathcal{I}, \mathcal{J}) \models \mathcal{T}_{12}$, and given a set \mathcal{X} of interpretations of Σ_1 , $\text{SAT}_{\mathcal{M}}(\mathcal{X})$ is defined as: $\text{SAT}_{\mathcal{M}}(\mathcal{X}) = \bigcup_{\mathcal{I} \in \mathcal{X}} \text{SAT}_{\mathcal{M}}(\mathcal{I})$.

Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a mapping, \mathcal{K}_1 a KB over Σ_1 , and \mathcal{K}_2 a KB over Σ_2 .

- \mathcal{K}_2 is called a *solution* for \mathcal{K}_1 under \mathcal{M} if $\text{MOD}(\mathcal{K}_2) \subseteq \text{SAT}_{\mathcal{M}}(\text{MOD}(\mathcal{K}_1))$, and
- \mathcal{K}_2 is called a *universal solution* for \mathcal{K}_1 under \mathcal{M} if $\text{MOD}(\mathcal{K}_2) = \text{SAT}_{\mathcal{M}}(\text{MOD}(\mathcal{K}_1))$.

Universal solutions present several limitations, as argued in [1, 2]. First, a universal solution (in $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ and $DL\text{-Lite}_{\mathcal{R}}$) does not always exist. Second, if it exists, then its TBox is trivial (that is, equivalent to the empty TBox). Finally, in the worst case the smallest universal solution is exponential in the size of the mapping and the source KB. A notion of solution parametrized w.r.t. a query language was proposed in [1, 2] in order to overcome these limitations. Such a notion, though weaker, is in line with the objective of (data and) KB exchange of providing in the target sufficient information to answer queries that could also be posed over the source.

Let \mathcal{Q} be a class of queries, $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ a mapping, $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ a KB over Σ_1 , and \mathcal{K}_2 a KB over Σ_2 . Then

- \mathcal{K}_2 is called a \mathcal{Q} -*solution* for \mathcal{K}_1 under \mathcal{M} if for every query $q \in \mathcal{Q}$ over Σ_2 , $\text{cert}(q, \langle \mathcal{T}_1 \cup \mathcal{T}_{12}, \mathcal{A}_1 \rangle) \subseteq \text{cert}(q, \mathcal{K}_2)$, and
- \mathcal{K}_2 is called a *universal \mathcal{Q} -solution* for \mathcal{K}_1 under \mathcal{M} if for every query $q \in \mathcal{Q}$ over Σ_2 , $\text{cert}(q, \langle \mathcal{T}_1 \cup \mathcal{T}_{12}, \mathcal{A}_1 \rangle) = \text{cert}(q, \mathcal{K}_2)$.

The definitions of solutions are illustrated in the following example.

Example 1. Assume $\Sigma_1 = \{\text{Painting}(\cdot), \text{PaintedBy}(\cdot, \cdot), \text{ArtMovement}(\cdot, \cdot)\}$ and $\Sigma_2 = \{\text{ArtPiece}(\cdot), \text{ArtAuthor}(\cdot, \cdot), \text{HasStyle}(\cdot, \cdot), \text{HasGenre}(\cdot, \cdot)\}$. Consider mapping $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, where \mathcal{T}_{12} is the following TBox:

$$\begin{array}{ll} \text{Painting} \sqsubseteq \text{ArtPiece} & \text{PaintedBy} \sqsubseteq \text{ArtAuthor} \\ \text{Painting} \sqsubseteq \exists \text{HasGenre} & \text{ArtMovement} \sqsubseteq \text{HasStyle} \end{array}$$

Further, assume $\mathcal{T}_1 = \{\text{Painting} \equiv \exists \text{PaintedBy}, \text{Painting} \sqsubseteq \exists \text{ArtMovement}\}$ and $\mathcal{A}_1 = \{\text{Painting}(\text{blacksquare})\}$. Then, a universal solution for the KB $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ under \mathcal{M} is the KB $\mathcal{K}_2 = \langle \mathcal{T}_2, \mathcal{A}_2 \rangle$, where $\mathcal{T}_2 = \emptyset$ and \mathcal{A}_2 is the following ABox, where $_n01$, $_n02$, and $_m01$ are labelled nulls:

$$\begin{array}{ll} \text{ArtPiece}(\text{blacksquare}) & \text{ArtAuthor}(\text{blacksquare}, _n01) \\ \text{HasGenre}(\text{blacksquare}, _m01) & \text{HasStyle}(\text{blacksquare}, _n02) \end{array}$$

Now, consider KB $\mathcal{K}'_2 = \langle \mathcal{T}'_2, \mathcal{A}'_2 \rangle$ with non-empty TBox, where $\mathcal{T}'_2 = \{\text{ArtPiece} \equiv \exists \text{ArtAuthor}, \text{ArtPiece} \sqsubseteq \exists \text{HasStyle}, \text{ArtPiece} \sqsubseteq \exists \text{HasGenre}\}$ and $\mathcal{A}'_2 = \{\text{ArtPiece}(\text{blacksquare})\}$. Then we have that \mathcal{K}'_2 is a solution for \mathcal{K}_1 under \mathcal{M} . However, we also have that \mathcal{K}'_2 is not a universal solution for \mathcal{K}_1 under \mathcal{M} . Notably, both KB \mathcal{K}_2 and KB \mathcal{K}'_2 are universal UCQ-solutions for KB \mathcal{K}_1 under mapping \mathcal{M} . ■

In order to understand the capacity of universal solutions, and also of the query-languages based notions of solutions to transfer implicit knowledge, the notion of *representability* has been introduced in [1, 2]. Here we adapt that definition to the case

where the KB is always satisfiable, as in $DL-Lite_{\mathcal{R}}^{pos}$. In the definition below we use $chase_{\mathcal{T}, \Sigma}(\mathcal{A})$ to denote the projection of $chase_{\mathcal{T}}(\mathcal{A})$ on the signature Σ .

Let \mathcal{L} be a DL, \mathcal{Q} a class of queries, $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ an \mathcal{L} -mapping, and \mathcal{T}_1 an \mathcal{L} -TBox over Σ_1 . Then,

- \mathcal{T}_1 is (\mathcal{Q} -)representable under \mathcal{M} if there exists an \mathcal{L} -TBox \mathcal{T}_2 over Σ_2 , called a (\mathcal{Q} -)representation of \mathcal{T}_1 under \mathcal{M} , such that for every ABox \mathcal{A}_1 over Σ_1 , $\langle \mathcal{T}_2, chase_{\mathcal{T}_{12}, \Sigma_2}(\mathcal{A}_1) \rangle$ is a (\mathcal{Q} -)universal solution for $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ under \mathcal{M} .
- \mathcal{T}_1 is weakly (\mathcal{Q} -)representable under \mathcal{M} if there exists a mapping $\mathcal{M}^* = (\Sigma_1, \Sigma_2, \mathcal{T}_{12}^*)$ such that $\mathcal{T}_{12} \subseteq \mathcal{T}_{12}^*$, $\mathcal{T}_1 \cup \mathcal{T}_{12} \models \mathcal{T}_{12}^*$, and \mathcal{T}_1 is (\mathcal{Q} -)representable under \mathcal{M}^* .

Example 2. Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ and \mathcal{T}_1 be as in Example 1. Then we have that $\mathcal{T}_2 = \{ArtPiece \sqsubseteq \exists ArtAuthor, ArtPiece \sqsubseteq \exists HasStyle, ArtPiece \sqsubseteq \exists HasGenre\}$ is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} .

On the other hand, if $\mathcal{M}' = (\Sigma_1, \Sigma_2, \mathcal{T}'_{12})$ with $\mathcal{T}'_{12} = \{PaintedBy \sqsubseteq ArtPiece\}$, then we have that \mathcal{T}_1 is not UCQ-representable under \mathcal{M}' : take ABox $\mathcal{A}_1 = \{Painting(\text{blacksquare})\}$, then $chase_{\mathcal{T}'_{12}, \Sigma_2}(\mathcal{A}_1) = \emptyset$ and for no TBox \mathcal{T}'_2 , $\langle \mathcal{T}'_2, chase_{\mathcal{T}'_{12}, \Sigma_2}(\mathcal{A}_1) \rangle$ is a universal UCQ-solution for $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ under \mathcal{M}' . However, if we consider $\mathcal{T}_{12}^* = \mathcal{T}'_{12} \cup \{Painting \sqsubseteq \exists ArtAuthor\}$, we conclude that \mathcal{T}_1 is weakly UCQ-representable under \mathcal{M}' since $\mathcal{T}'_{12} \subseteq \mathcal{T}_{12}^*$, $\mathcal{T}_1 \cup \mathcal{T}'_{12} \models \mathcal{T}_{12}^*$ and \mathcal{T}_1 is UCQ-representable under $\mathcal{M}^* = (\Sigma_1, \Sigma_2, \mathcal{T}_{12}^*)$ (in fact, \emptyset is a UCQ-representation of \mathcal{T}_1 under \mathcal{M}^*). ■

3 Solving UCQ-Representability for $DL-Lite_{\mathcal{R}}^{pos}$

In this section, we show that the UCQ-representability problem can be solved in polynomial time for the case where TBoxes and mappings are expressed in $DL-Lite_{\mathcal{R}}^{pos}$. More specifically, we give a polynomial time algorithm $UCQREP^{pos}$ that, given a $DL-Lite_{\mathcal{R}}^{pos}$ -mapping $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ and a $DL-Lite_{\mathcal{R}}^{pos}$ -TBox \mathcal{T}_1 , verifies whether \mathcal{T}_1 is UCQ-representable under \mathcal{M} , and if this is the case computes a UCQ-representation of \mathcal{T}_1 under \mathcal{M} . Moreover, we also show that this algorithm can be used to solve the UCQ-representability problem for the case of $DL-Lite_{\mathcal{RDFS}}$. It is important to notice that the algorithm we present can be used to compute universal UCQ-solutions of polynomial size, which make good use of the source implicit knowledge. Thus, this algorithm computes solutions with good properties to be used in practice.

A related problem is that of query inseparability [7], which can be formulated as follows: given TBoxes \mathcal{T}_1 and \mathcal{T}_2 , and a signature Σ , decide whether for each ABox \mathcal{A} over Σ and for each query q over Σ , $cert(q, \langle \mathcal{T}_1, \mathcal{A} \rangle) = cert(q, \langle \mathcal{T}_2, \mathcal{A} \rangle)$. In contrast to our polynomial result for UCQ-representability, query inseparability has been proved to be PSPACE-hard for $DL-Lite_{\mathcal{R}}$ TBoxes and CQs [7], and an analysis of the proof shows that the same lower bound holds already for $DL-Lite_{\mathcal{R}}^{pos}$.

3.1 Checking Whether a Given Target TBox is a UCQ-Representation

We start by considering the decision problem associated with UCQ-representability: Given a $DL-Lite_{\mathcal{R}}$ -mapping $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, a $DL-Lite_{\mathcal{R}}$ -TBox \mathcal{T}_1 over Σ_1 , and

a $DL\text{-Lite}_{\mathcal{R}}$ -TBox \mathcal{T}_2 over Σ_2 , check whether \mathcal{T}_2 is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} , i.e., for each ABox \mathcal{A}_1 over Σ_1 , $\langle \mathcal{T}_2, \text{chase}_{\mathcal{T}_2, \Sigma_2}(\mathcal{A}_1) \rangle$ is a universal UCQ-solution for $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ under \mathcal{M} . This problem can be solved in two steps:

- (C1) Check whether for each ABox \mathcal{A}_1 over Σ_1 , $\langle \mathcal{T}_2, \text{chase}_{\mathcal{T}_2, \Sigma_2}(\mathcal{A}_1) \rangle$ is a UCQ-solution for $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ under \mathcal{M} .
- (C2) Check whether for each ABox \mathcal{A}_1 over Σ_1 and for each UCQ q over Σ_2 , we have that $\text{cert}(q, \langle \mathcal{T}_2, \text{chase}_{\mathcal{T}_2, \Sigma_2}(\mathcal{A}_1) \rangle) \subseteq \text{cert}(q, \langle \mathcal{T}_1 \cup \mathcal{T}_{12}, \mathcal{A}_1 \rangle)$.

If both checks succeed, then \mathcal{T}_2 is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} , otherwise not. We develop now techniques to perform these two checks in polynomial time.

Checking Condition (C1). For a $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ TBox \mathcal{T} and a concept or role N , we define the *upward closure of N w.r.t. \mathcal{T}* as the set $\mathbb{U}_{\mathcal{T}}(N) = \{N' \mid N' \text{ is concept or role and } \mathcal{T} \models N \sqsubseteq N'\}$, and the *strict upward closure* $\mathbb{S}_{\mathcal{T}}(N)$ as $\mathbb{U}_{\mathcal{T}}(N) \setminus \{N\}$. Then, for a set \mathbf{N} of concepts and roles we define $\mathbb{U}_{\mathcal{T}}(\mathbf{N}) = \bigcup_{N \in \mathbf{N}} \mathbb{U}_{\mathcal{T}}(N)$, and its strict version $\mathbb{S}_{\mathcal{T}}(\mathbf{N})$. Notice that both $\mathbb{S}_{\mathcal{T}_2}(\mathbb{U}_{\mathcal{T}_1}(N))$ and $\mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(N))$ are sets over Σ_2 , for each concept or role N over Σ_1 .

With these notions in place, we can provide a necessary and sufficient condition for the satisfaction of condition (C1).

Proposition 1. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ -mapping, \mathcal{T}_1 a $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ -TBox over Σ_1 , and \mathcal{T}_2 a $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ -TBox over Σ_2 . Then \mathcal{T}_1 , \mathcal{T}_2 , and \mathcal{M} satisfy condition (C1) iff the following conditions are satisfied:*

- (A) $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(B)) \subseteq \mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(B))$, for each basic concept B over Σ_1 ;
- (B) $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(R)) \subseteq \mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(R))$, for each basic role R over Σ_1 ;
- (C) for each basic concept B and each basic role R over Σ_1 such that $\exists R \in \mathbb{U}_{\mathcal{T}_1}(B)$ and $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(\exists R^-)) \neq \emptyset$, we have that
 - if $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(R)) \neq \emptyset$, then there exists a role $Q_{R,B}$ over Σ_2 such that
 - (CA) $\exists Q_{R,B} \in \mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(B))$,
 - (CB) $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(R)) \subseteq \mathbb{U}_{\mathcal{T}_2}(Q_{R,B})$, and
 - (CC) $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(\exists R^-)) \subseteq \mathbb{U}_{\mathcal{T}_2}(\exists Q_{R,B}^-)$,
 - and if $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(R)) = \emptyset$, either
 - (CD) $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(\exists R^-)) \subseteq \mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(B))$,
 - or there exist roles $Q_{R,B}^1, \dots, Q_{R,B}^n$ over Σ_2 such that
 - (CE) $\exists Q_{R,B}^1 \in \mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(B))$,
 - (CF) $\mathcal{T}_2 \models \exists(Q_{R,B}^1)^- \sqsubseteq \exists Q_{R,B}^2, \dots, \exists(Q_{R,B}^{n-1})^- \sqsubseteq \exists Q_{R,B}^n$, and
 - (CG) $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(\exists R^-)) \subseteq \mathbb{U}_{\mathcal{T}_2}(\exists(Q_{R,B}^n)^-)$.

It is important to notice that the necessary and sufficient condition in Proposition 1 can be checked in polynomial time, as the implication problem for $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ can be solved in polynomial time. In particular, for a basic concept B and a basic role R over Σ_1 such that $\exists R \in \mathbb{U}_{\mathcal{T}_1}(B)$, $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(\exists R^-)) \neq \emptyset$, $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(R)) = \emptyset$, and $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(\exists R^-)) \not\subseteq \mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(B))$, checking the existence of roles $Q_{R,B}^1, \dots, Q_{R,B}^n$ over Σ_2 satisfying conditions (CE), (CF), and (CG) can be reduced to checking reachability in a directed graph. Indeed, for each pair of basic concepts B_2, B'_2 over Σ_2

such that $B_2 \in \mathbb{S}_{\mathcal{M}}(B)$ and $\mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(\exists R^-)) \subseteq \mathbb{U}_{\mathcal{T}_2}(B'_2)$, we use the following approach to check for the existence of the roles $Q_{R,B}^1, \dots, Q_{R,B}^n$ over Σ_2 such that $\mathcal{T}_2 \models B_2 \sqsubseteq \exists Q_{R,B}^1, \mathcal{T}_2 \models \exists(Q_{R,B}^1)^- \sqsubseteq \exists Q_{R,B}^2, \dots, \mathcal{T}_2 \models \exists(Q_{R,B}^n)^- \sqsubseteq B'_2$. Let $G = (V, E)$ be the directed graph defined as:

$$\begin{aligned} V &= \{B_2, B'_2\} \cup \{Q_2 \mid Q_2 \text{ is a role in } \Sigma_2\} \\ E &= \{(B_2, B'_2) \mid \mathcal{T}_2 \models B_2 \sqsubseteq B'_2\} \cup \{(B_2, Q_2) \mid \mathcal{T}_2 \models B_2 \sqsubseteq \exists Q_2\} \cup \\ &\quad \{(Q_2, B'_2) \mid \mathcal{T}_2 \models \exists Q_2^- \sqsubseteq B'_2\} \cup \{(Q_2, Q'_2) \mid \mathcal{T}_2 \models \exists Q_2^- \sqsubseteq \exists Q'_2\} \end{aligned}$$

Then we test for the existence of the roles $Q_{R,B}^1, \dots, Q_{R,B}^n$ by verifying whether B'_2 is reachable from B_2 in G . If for some pair B_2, B'_2 the aforementioned two-step test succeed, then we have that there exist roles $Q_{R,B}^1, \dots, Q_{R,B}^n$ that satisfy conditions (CE), (CF), and (CG). Otherwise, we know that such roles do not exist.

Checking Condition (C2). We rely on the following result:

Proposition 2. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -mapping, \mathcal{T}_1 a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -TBox over Σ_1 , and \mathcal{T}_2 a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -TBox over Σ_2 . Then $\mathcal{T}_1, \mathcal{T}_2$, and \mathcal{M} satisfy condition (C2) iff the following conditions are satisfied:*

- (A) $\mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(B)) \subseteq \mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(B))$ for each basic concept B over Σ_1 ;
- (B) $\mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(R)) \subseteq \mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(R))$ for each role $R \in \Sigma_1$;
- (C) for each basic role Q over Σ_2 and each basic concept B over Σ_1 such that $\exists Q \in \mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(B))$ and $\mathbb{U}_{\mathcal{T}_2}(\exists Q^-) \neq \{\exists Q^-\}$, there exists a role $R_{Q,B}$ over Σ_1 s.t.
 - (CA) $\exists R_{Q,B} \in \mathbb{U}_{\mathcal{T}_1}(B)$ and
 - (CB) $Q \in \mathbb{S}_{\mathcal{M}}(R_{Q,B})$.

The necessary and sufficient condition in Proposition 2 can be checked in polynomial time, as the implication problem can be solved in polynomial time for $DL\text{-Lite}_{\mathcal{R}}^{pos}$.

Thus, given that, by Propositions 1 and 2, both conditions (C1) and (C2) can be tested in polynomial time, we obtain the following result.

Theorem 1. *The problem of verifying, given a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -mapping $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -TBox \mathcal{T}_1 over Σ_1 , and a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -TBox \mathcal{T}_2 over Σ_2 , whether \mathcal{T}_2 is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} can be solved in polynomial time.*

3.2 The Algorithm for Computing a UCQ-Representation

In what follows, we present the algorithm $UCQREP^{pos}$, which verifies whether a source TBox \mathcal{T}_1 is UCQ-representable under a mapping \mathcal{M} , and if this is the case returns a UCQ-representation of \mathcal{T}_1 under \mathcal{M} .

Intuitively, given a source TBox \mathcal{T}_1 and a mapping \mathcal{M} , algorithm $UCQREP^{pos}$ constructs the best possible candidate \mathcal{T}_2 for a UCQ-representation of \mathcal{T}_1 under \mathcal{M} (given the conditions in Propositions 1 and 2), and then checks whether \mathcal{T}_2 effectively satisfies the properties required for a UCQ-representation (note, that \mathcal{T}_2 is indeed a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ TBox). To prove the correctness of this algorithm, we need to show that \mathcal{T}_1 is UCQ-representable under \mathcal{M} if and only if \mathcal{T}_2 is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} . This

Algorithm: UCQREP^{pos}($\mathcal{T}_1, \mathcal{M}$)

Input: A $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ -mapping $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ and a $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ -TBox \mathcal{T}_1 over Σ_1 .

Output: A $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ -TBox \mathcal{T}_2 over Σ_2 that is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} , if \mathcal{T}_1 is UCQ-representable under \mathcal{M} . The keyword *false* otherwise.

1. Let \mathcal{T}_2 be a TBox over Σ_2 defined as:

$$\mathcal{T}_2 = \{N_2 \sqsubseteq M_2 \mid N_1 \text{ a basic concept or role over } \Sigma_1, \\ N_2 \in \mathbb{S}_{\mathcal{M}}(N_1), M_2 \in \mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(N_1))\}$$

2. Remove from \mathcal{T}_2 every inclusion $N_2 \sqsubseteq M_2$ such that (i) $N_2 \in \mathbb{S}_{\mathcal{M}}(N_1)$ for some N_1 over Σ_1 , and (ii) for every M_1 over Σ_1 such that $M_2 \in \mathbb{S}_{\mathcal{M}}(M_1)$, it holds that $\mathcal{T}_1 \not\models N_1 \sqsubseteq M_1$. Moreover, if $N_2 = \exists R_2$ and $M_2 = \exists R'_2$, then also remove inclusions $R_2 \sqsubseteq R'_2$ and $R_2^- \sqsubseteq R'^-_2$ from \mathcal{T}_2 .
3. Remove from \mathcal{T}_2 every inclusion of the form either $\exists R_2^- \sqsubseteq B_2$ or $R_2 \sqsubseteq R'_2$ or $R_2^- \sqsubseteq R'^-_2$ for roles R_2, R'_2 and a concept B_2 over Σ_2 , if there exists a concept B_1 over Σ_1 such that (i) $\exists R_2 \in \mathbb{S}_{\mathcal{M}}(B_1)$, and (ii) for every role R_1 over Σ_1 such that $\exists R_1 \in \mathbb{U}_{\mathcal{T}_1}(B_1)$ and $R_2 \in \mathbb{S}_{\mathcal{M}}(R_1)$, it holds that $\mathcal{T}_1 \not\models B_1 \sqsubseteq \exists R_1$.
4. Verify whether \mathcal{T}_2 is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} . If the test succeeds, return \mathcal{T}_2 , otherwise return *false*.

Fig. 1. Algorithm to compute the UCQ-representation of a $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ TBox \mathcal{T}_1 under a $DL\text{-Lite}_{\mathcal{R}}^{\text{pos}}$ mapping \mathcal{M} .

is done in the following theorem, where it is also proved that the algorithm works in polynomial time. The latter is a consequence of the fact that \mathcal{T}_2 is of polynomial size in the sizes of \mathcal{T}_1 and \mathcal{M} , and that, by Theorem 1, it is possible to check in polynomial time whether \mathcal{T}_2 is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} .

Theorem 2. *Algorithm UCQREP^{pos} is correct and runs in polynomial time.*

The following examples illustrate how algorithm UCQREP^{pos} works.

Example 3. Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, where $\Sigma_1 = \{A_1(\cdot), B_1(\cdot), C_1(\cdot)\}$, $\Sigma_2 = \{A_2(\cdot), B_2(\cdot)\}$, and $\mathcal{T}_{12} = \{A_1 \sqsubseteq A_2, B_1 \sqsubseteq B_2, C_1 \sqsubseteq B_2\}$. Furthermore, assume that $\mathcal{T}_1 = \{B_1 \sqsubseteq A_1\}$. Then, in step 1, the algorithm constructs the TBox $\mathcal{T}_2 = \{B_2 \sqsubseteq A_2\}$. In step 2, it removes the only axiom from \mathcal{T}_2 as $B_2 \in \mathbb{S}_{\mathcal{M}}(C_1)$ and $\mathcal{T}_1 \not\models C_1 \sqsubseteq A_1$. In step 3, it does nothing, and finally, at the last step it checks whether the empty TBox \mathcal{T}_2 is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} . Since $A_2 \in \mathbb{S}_{\mathcal{M}}(\mathbb{U}_{\mathcal{T}_1}(B_1))$ and $A_2 \notin \mathbb{U}_{\mathcal{T}_2}(\mathbb{S}_{\mathcal{M}}(B_1))$, the algorithm returns *false*. ■

Example 4. Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, where $\Sigma_1 = \{B_1(\cdot), P_1(\cdot, \cdot), R_1(\cdot, \cdot)\}$, $\Sigma_2 = \{A_2(\cdot), B_2(\cdot), R_2(\cdot, \cdot)\}$, and $\mathcal{T}_{12} = \{\exists P_1^- \sqsubseteq A_2, B_1 \sqsubseteq B_2, R_1 \sqsubseteq R_2\}$. Furthermore, assume that $\mathcal{T}_1 = \{B_1 \sqsubseteq \exists P_1, B_1 \sqsubseteq \exists R_1, \exists R_1^- \sqsubseteq \exists P_1^-\}$. Then, in step 1, the algorithm constructs the TBox $\mathcal{T}_2 = \{B_2 \sqsubseteq \exists R_2, \exists R_2^- \sqsubseteq A_2\}$. It does not remove anything in steps 2 and 3. Finally, at the last step it successfully checks that \mathcal{T}_2 is a UCQ-representation of \mathcal{T}_1 under \mathcal{M} and returns \mathcal{T}_2 . ■

3.3 Solving UCQ-Representability for $DL\text{-Lite}_{RDFS}$

It is not difficult to see that if the input of algorithm $UCQREP^{pos}$ is a $DL\text{-Lite}_{RDFS}$ -mapping $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ and a $DL\text{-Lite}_{RDFS}$ -TBox \mathcal{T}_1 over Σ_1 , then the set \mathcal{T}_2 computed by this algorithm is a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -TBox over Σ_2 that can be easily transformed into an equivalent $DL\text{-Lite}_{RDFS}$ -TBox. Indeed, \mathcal{T}_2 might contain inclusions between basic concepts of the form $\exists R_2 \sqsubseteq \exists R'_2$, but this occurs only if \mathcal{T}_2 implies also the role inclusion $R_2 \sqsubseteq R'_2$. Hence, all concept inclusions that would fall outside $DL\text{-Lite}_{RDFS}$ are implied by the $DL\text{-Lite}_{RDFS}$ fragment of \mathcal{T}_2 and can be removed from \mathcal{T}_2 without affecting its semantics. Thus, we conclude that algorithm $UCQREP^{pos}$ can also be used to solve in polynomial time the UCQ-representability problem for $DL\text{-Lite}_{RDFS}$ mappings and TBoxes.

4 Solving Weak UCQ-Representability for $DL\text{-Lite}_{\mathcal{R}}^{pos}$

In this section, we show that also the weak UCQ-representability problem can be solved in polynomial time when TBoxes and mappings are expressed $DL\text{-Lite}_{\mathcal{R}}^{pos}$. We first need to introduce some terminology. Given a $DL\text{-Lite}_{\mathcal{R}}$ -TBox \mathcal{T} over a signature Σ and a UCQ q over Σ , a UCQ q_r over Σ is said to be a *perfect reformulation* of q w.r.t. \mathcal{T} if for every ABox \mathcal{A} over Σ , it holds that [6]: $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = cert(q_r, \langle \emptyset, \mathcal{A} \rangle)$. That is, the certain answers to the UCQ q over a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ can be computed by posing the UCQ q_r over the ABox \mathcal{A} . It is well-known that every UCQ q admits a perfect reformulation w.r.t. a $DL\text{-Lite}_{\mathcal{R}}$ -TBox \mathcal{T} , which can be computed in polynomial time [6].

Interestingly, the fundamental notion of perfect reformulation can be used to solve the UCQ-representability problem for $DL\text{-Lite}_{\mathcal{R}}^{pos}$. More precisely, let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a $DL\text{-Lite}_{\mathcal{R}}$ -mapping and \mathcal{T}_1 a $DL\text{-Lite}_{\mathcal{R}}$ -TBox over Σ_1 . Then define a mapping $COMP(\mathcal{M}, \mathcal{T}_1) = (\Sigma_1, \Sigma_2, \mathcal{T}_{12}^*)$ that extends \mathcal{M} by compiling the knowledge from \mathcal{T}_1 into \mathcal{T}_{12} . Formally, for a basic concept or role N over Σ_1 , let bq_N be the CQ defined as follows: $bq_A(x) = A(x)$, $bq_{\exists P}(x) = \exists y.P(x, y)$, $bq_{\exists P^-}(x) = \exists y.P(y, x)$, $bq_P(x, y) = P(x, y)$, and $bq_{P^-}(x, y) = P(y, x)$. Then, for every concept inclusion $B \sqsubseteq C \in \mathcal{T}_{12}$ and for every CQ q in the perfect reformulation of bq_B w.r.t. \mathcal{T}_1 , include $C_q \sqsubseteq C$ into \mathcal{T}_{12}^* , where C_q is the (unique) basic concept such that $bq_{C_q} = q$. Also, for every role inclusion $R \sqsubseteq Q \in \mathcal{T}_{12}$ and for every CQ q in the perfect reformulation of bq_R w.r.t. \mathcal{T}_1 , include $R_q \sqsubseteq Q$ into \mathcal{T}_{12}^* , where R_q is the basic role such that $bq_{R_q} = q$.

It is important to notice that if $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ is a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -mapping and \mathcal{T}_1 a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -TBox over Σ_1 , then $COMP(\mathcal{M}, \mathcal{T}_1) = (\Sigma_1, \Sigma_2, \mathcal{T}_{12}^*)$ is a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -mapping that can be computed in polynomial time in the sizes of \mathcal{M} and \mathcal{T}_1 . Therefore, given that the set of inclusions defining $COMP(\mathcal{M}, \mathcal{T}_1)$ contains the set of inclusions defining \mathcal{M} and $\mathcal{T}_1 \cup \mathcal{T}_{12} \models \mathcal{T}_{12}^*$, we conclude that $COMP(\mathcal{M}, \mathcal{T}_1)$ can be used to check in polynomial time whether \mathcal{T}_1 is weakly UCQ-representable under \mathcal{M} .

Theorem 3. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -mapping and \mathcal{T}_1 a $DL\text{-Lite}_{\mathcal{R}}^{pos}$ -TBox over Σ_1 . Then \mathcal{T}_1 is weakly UCQ-representable under \mathcal{M} if and only if \mathcal{T}_1 is UCQ-representable under $COMP(\mathcal{M}, \mathcal{T}_1)$.*

From this result and Theorem 2 we obtain a polynomial time algorithm for solving the weak UCQ-representability problem for $DL\text{-Lite}_{\mathcal{R}}^{pos}$ mappings and TBoxes.

The example below shows a $DL-Lite_{\mathcal{R}}^{pos}$ TBox \mathcal{T}_1 and a $DL-Lite_{\mathcal{R}}^{pos}$ mapping \mathcal{M} such that \mathcal{T}_1 is not weakly UCQ-representable under \mathcal{M} .

Example 5. Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, where $\Sigma_1 = \{P_1(\cdot, \cdot), B_1(\cdot)\}$, $\Sigma_2 = \{A_2(\cdot), B_2(\cdot)\}$, and $\mathcal{T}_{12} = \{B_1 \sqsubseteq B_2, \exists P_1^- \sqsubseteq A_2\}$. Furthermore, assume that $\mathcal{T}_1 = \{B_1 \sqsubseteq \exists P_1^-\}$. Then \mathcal{T}_1 is not weakly UCQ-representable under \mathcal{M} . In fact, given that the perfect reformulation of $\exists P_1^-$ w.r.t. TBox \mathcal{T}_1 is $\exists P_1^-$ itself, and likewise for concept B_1 , we have that $\mathcal{M} = \text{COMP}(\mathcal{M}, \mathcal{T}_1)$ and, thus, \mathcal{T}_1 is not weakly UCQ-representable under \mathcal{M} , as \mathcal{T}_1 is not UCQ-representable under \mathcal{M} . ■

Instead, as shown in [1, 2], for each $DL-Lite_{RDFS}$ TBox \mathcal{T}_1 and $DL-Lite_{RDFS}$ mapping \mathcal{M} , \mathcal{T}_1 is weakly UCQ-representable under \mathcal{M} .

5 Conclusions

In this paper, we have extended previous results on representability in the knowledge exchange framework to $DL-Lite_{\mathcal{R}}^{pos}$, a DL of the $DL-Lite$ family that allows for existentials in the right-hand side of inclusion assertions, both in the source TBox and in the mapping. We are currently working on extending our results to $DL-Lite_{\mathcal{R}}$, which includes disjointness assertions, and to the other DLs in the extended $DL-Lite$ family [4]. A further interesting problem that we are investigating is that of checking the existence of universal solutions for $DL-Lite_{\mathcal{R}}^{pos}$ and other more expressive DLs.

Acknowledgements. The authors were supported by the EU Marie Curie action FP7-PEOPLE-2009-IRSES (grant 24761), by Fondecyt (grant 1090565), by the EU ICT projects ACSI (grant FP7-257593) and FOX (grant FP7-233599), and by the NWO project DEX (612.001.012).

References

1. Arenas, M., Botoeva, E., Calvanese, D.: Knowledge base exchange. In: Proc. of DL 2011. CEUR, ceur-ws.org, vol. 745 (2011)
2. Arenas, M., Botoeva, E., Calvanese, D., Ryzhikov, V., Sherkhonov, E.: Exchanging description logic knowledge bases. In: Proc. of KR 2012 (2012)
3. Arenas, M., Pérez, J., Reutter, J.L.: Data exchange beyond complete data. In: Proc. of PODS 2011. pp. 83–94 (2011)
4. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: The $DL-Lite$ family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
5. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium (Feb 2004), available at <http://www.w3.org/TR/rdf-schema/>
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The $DL-Lite$ family. J. of Automated Reasoning 39(3), 385–429 (2007)
7. Konev, B., Kontchakov, R., Ludwig, M., Schneider, T., Wolter, F., Zakharyashev, M.: Conjunctive query inseparability of OWL 2 QL TBoxes. In: Proc. of AAAI 2011. pp. 221–226 (2011)

Modular Combination of Reasoners for Ontology Classification

Ana Armas Romero, Bernardo Cuenca Grau, Ian Horrocks

Department of Computer Science. University of Oxford

Abstract. Classification is a fundamental reasoning task in ontology design, and there is currently a wide range of reasoners highly optimised for classification of *SR_{OIQ}* ontologies. Existing reasoners, however, do not exploit the fact that most of the axioms in many realistic *SR_{OIQ}* ontologies are expressed in some lightweight DL, such as \mathcal{EL}^{++} . In this paper, we propose a novel reasoning technique that allows us to completely classify a large subset of the signature of a *SR_{OIQ}* ontology by relying only on a reasoner for a given lightweight DL. We also show how this information can then be exploited by the fully-fledged *SR_{OIQ}* reasoner HermiT to complete the classification of the ontology.

1 Introduction

Classification —the problem of identifying the subsumption relationships between all pairs of atomic concepts occurring in the input ontology— is a fundamental reasoning task in ontology design. The decision problems associated to classification, however, have a very high worst-case complexity for expressive DLs; in particular, subsumption w.r.t. an ontology is 2NEXPTIME-complete for *SR_{OIQ}* [14] —the DL underlying the standard ontology language OWL 2 [5].

Despite these discouraging complexity results, considerable effort has been devoted to making classification feasible in practice. As a result, many reasoning algorithms and optimisation techniques have been developed, and there is currently a wide range of highly-optimised reasoners, such as Pellet [19], FaCT++ [20], RacerPro [9] and HermiT [6], that support classification of ontologies written in expressive description logics.

Since individual subsumption tests performed during classification can be computationally very expensive, most DL reasoners implement variants of the well-known Enhanced Traversal Algorithm [2], which reduces the number of required subsumption tests. Sophisticated optimisation techniques are also implemented on top of these algorithms to further reduce the number of potentially expensive subsumption tests [10].

A widely implemented technique is the *told subsumptions* optimisation [10], which provides an inexpensive way of computing subsumption relationships that hold in the input ontology. In typical ontologies, however, most candidate subsumption relationships between atomic concepts will not hold; hence, efficiently identifying and exploiting such *non-subsumption* relationships becomes critical

in practice, and several optimisation techniques have been developed with this goal in mind. In particular, the *completely defined concepts* optimisation [21] identifies a fragment of the ontology for which *told* subsumption provides complete information; furthermore *model-merging* and other related techniques exploit the computations performed during individual concept satisfiability tests to detect non-subsumptions [10, 8, 6]. However, although these techniques have proved effective in practice, the classification of very large ontologies can still require a large number of expensive subsumption tests.

In recent years, there has been a growing interest in so-called *lightweight DLs*. The description logic \mathcal{EL}^{++} [1], for example, can capture several prominent ontologies, and allows classification to be performed in polynomial time. Reasoners specifically designed for \mathcal{EL}^{++} , such as CEL [3] and ELK [15], can classify ontologies as large as SNOMED CT in a few seconds.

Unfortunately, many ontologies fall outside the \mathcal{EL}^{++} fragment, and so cannot be classified using \mathcal{EL}^{++} reasoners. In many cases, however, such ontologies contain only a relatively small number of non \mathcal{EL}^{++} axioms. For example, out of the 219,224 axioms in the latest version of NCI, only 65 are non \mathcal{EL}^{++} . Being able to use an \mathcal{EL}^{++} reasoner to efficiently compute most of the subsumptions and non-subsumptions required to classify these ontologies could lead to significant improvements in both performance and scalability.

In this paper, we propose a technique where a reasoner for some DL \mathcal{L} is used as “black box” by a reasoner for a more expressive logic \mathcal{L}' . We focus on the case where \mathcal{L}' is *SROIQ*, and we present a classification algorithm that, given a *SROIQ* ontology \mathcal{O} , proceeds as follows:

1. It computes a signature $\Sigma^{\mathcal{L}} \subseteq \text{Sig}(\mathcal{O})$ and a fragment $\mathcal{M}^{\mathcal{L}} \subseteq \mathcal{O}$ written in \mathcal{L} such that the concepts in $\Sigma^{\mathcal{L}}$ can be completely classified using only the axioms in $\mathcal{M}^{\mathcal{L}}$; more precisely, $\Sigma^{\mathcal{L}}$ and $\mathcal{M}^{\mathcal{L}}$ will be such that, for each atomic concept $A \in \Sigma^{\mathcal{L}}$ and each $B \in \text{Sig}(\mathcal{O}) \cup \{\top, \perp\}$, we have $\mathcal{O} \models A \sqsubseteq B$ iff $\mathcal{M}^{\mathcal{L}} \models A \sqsubseteq B$.
2. It classifies $\mathcal{M}^{\mathcal{L}}$ using an \mathcal{L} -reasoner and feeds (in a compact way) the obtained (non-)subsumptions to a *SROIQ*-reasoner, such as HermiT, that can effectively exploit this information [6].

Step 1 involves two important technical challenges. First, $\Sigma^{\mathcal{L}}$ should be as large as possible; in particular, for ontologies with only a few non- \mathcal{L} axioms, it is reasonable to expect $\Sigma^{\mathcal{L}}$ to contain most of the ontology’s signature. Second, $\mathcal{M}^{\mathcal{L}}$ must be *complete* for $\Sigma^{\mathcal{L}}$. Although techniques such as the completely defined concepts optimisation can be used to identify a complete fragment, these techniques are very restricted; thus, we exploit module extraction techniques [4, 7], which, in addition to giving completeness guarantees, are more generally applicable, more flexible, and more robust.

We believe that our results are interesting from both a theoretical and a practical point of view. We show that given a *SROIQ* ontology \mathcal{O} that is not captured by any known polynomial fragment of *SROIQ*, it is often possible to identify a large subset Σ of $\text{Sig}(\mathcal{O})$ such that all subsumers of concepts in Σ w.r.t. \mathcal{O} can be computed using a polynomial time classification algorithm. From

a practical point of view, our first experiments with a prototype implementation suggest the potential of this approach for optimising classification.

This paper is supplemented by an online Appendix containing additional technical details.¹

2 Preliminaries

We adopt standard DL notation, as well as standard notions of signature, interpretations, entailment, satisfiability and subsumption. We also assume basic familiarity with the description logics \mathcal{SROIQ} [11] and \mathcal{EL}^{++} [1]. When talking about *ontologies* and *axioms* we will implicitly refer to \mathcal{SROIQ} -ontologies and \mathcal{SROIQ} -axioms, respectively.

We denote with $\text{Sig}(\mathcal{O})$ (respectively, $\text{Sig}(\alpha)$) the signature of an ontology \mathcal{O} (respectively, of an axiom α). Furthermore, given an ontology \mathcal{O} and a DL $\mathcal{L} \subseteq \mathcal{SROIQ}$, we denote with $\mathcal{O}_{\mathcal{L}}$ the subset of \mathcal{L} -axioms in \mathcal{O} .

2.1 Module Extraction

Intuitively, a module \mathcal{M} for an ontology \mathcal{O} w.r.t. a signature Σ is an ontology $\mathcal{M} \subseteq \mathcal{O}$ such that \mathcal{M} entails the same axioms over Σ as \mathcal{O} .

This intuition is typically formalised using different notions of a *conservative extension* [16, 4]. In this paper, we define modules in terms of a model-theoretic notion of conservative extension.

Definition 1 (Model Conservative Extension). *Let \mathcal{O} be an ontology and let $\Sigma \subseteq \text{Sig}(\mathcal{O})$. We say that \mathcal{O} is a model conservative extension of $\mathcal{M} \subseteq \mathcal{O}$ w.r.t. Σ if, for every model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of \mathcal{M} , there exists a model $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of \mathcal{O} such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $X^{\mathcal{I}} = X^{\mathcal{J}}$ for every symbol $X \in \Sigma$.*

Definition 2 (Module). *Let \mathcal{O} be an ontology and let Σ be a signature. We say that $\mathcal{M} \subseteq \mathcal{O}$ is a module in \mathcal{O} w.r.t. Σ if \mathcal{O} is a model conservative extension of \mathcal{M} w.r.t. Σ .*

In particular, if \mathcal{M} is a module in \mathcal{O} w.r.t. Σ , then the following condition holds: for each axiom α with $\text{Sig}(\alpha) \subseteq \Sigma$, we have $\mathcal{M} \models \alpha$ iff $\mathcal{O} \models \alpha$.

The problem of checking whether \mathcal{M} is a module in \mathcal{O} w.r.t. Σ , however, is already undecidable for \mathcal{EL}^{++} [17], so approximations are typically needed in practice. The following sufficient condition for model conservativity is known to work well in practice [4].

Definition 3 (\emptyset -locality). *Let Σ be a signature and let \mathcal{O} be an ontology. An interpretation \mathcal{I} is \emptyset -local for Σ if for every atomic concept $A \notin \Sigma$ and every atomic role $R \notin \Sigma$, we have $A^{\mathcal{I}} = R^{\mathcal{I}} = \emptyset$. An axiom α is \emptyset -local for Σ if $\mathcal{I} \models \alpha$ for each \mathcal{I} that is \emptyset -local for Σ . An ontology \mathcal{O} is \emptyset -local for Σ if every axiom in \mathcal{O} is \emptyset -local for Σ .*

¹ <http://www.cs.ox.ac.uk/files/4770/ModClassDL12.pdf>

Checking \emptyset -locality for $SR\mathcal{OIQ}$ axioms is, however, a PSPACE-complete problem [4]. Since our goal is to optimise classification, checking \emptyset -locality might still be too costly. Instead, we will use \perp -locality — a well-known sufficient syntactic condition for \emptyset -locality which has been successfully used for both ontology reuse and reasoning problems [4, 12, 18, 7].

The precise grammar defining \perp -locality for $SR\mathcal{OIQ}$ is given for reference in the Appendix, and can also be found in the literature [7, 4]. It suffices to consider that, for each \mathcal{O} and Σ , \perp -locality implies \emptyset -locality and it can be checked in polynomial time. Furthermore, the following property holds [7, 4]:

Proposition 1. *If an axiom α is \perp -local w.r.t. a signature Σ , then α is \perp -local w.r.t. Σ' for any $\Sigma' \subseteq \Sigma$.*

We can use \perp -locality to define the notion of a \perp -module. The aforementioned properties of \perp -locality ensure that, if \mathcal{M} is a \perp -module w.r.t. Σ in \mathcal{O} as defined next, then it is also a module w.r.t. Σ in \mathcal{O} .

Definition 4 (\perp -module). *An ontology $\mathcal{M} \subseteq \mathcal{O}$ is a \perp -module in \mathcal{O} w.r.t. Σ if $\mathcal{O} \setminus \mathcal{M}$ is \perp -local for $\Sigma \cup \text{Sig}(\mathcal{M})$.*

Clearly, there is a unique smallest \perp -module for a given \mathcal{O} and Σ (the smallest subset $\mathcal{M} \subseteq \mathcal{O}$ s.t. $\mathcal{O} \setminus \mathcal{M}$ is \perp -local for $\Sigma \cup \text{Sig}(\mathcal{M})$). In what follows, we refer to such smallest module as *the* \perp -module in \mathcal{O} w.r.t. Σ and we denote it $\mathcal{M}_{[\mathcal{O}, \Sigma]}$.

In addition to being modules as in Definition 2, \perp -modules also enjoy an additional property that makes them especially well-suited for optimising ontology classification [7].

Proposition 2. *Let \mathcal{O} be an ontology, let A, B be concepts in $\text{Sig}(\mathcal{O}) \cup \{\top, \perp\}$, let $\Sigma \subseteq \text{Sig}(\mathcal{O})$ with $A \in \Sigma$, and let $\mathcal{M} \subseteq \mathcal{O}$ be a \perp -module in \mathcal{O} w.r.t. Σ . Then $\mathcal{O} \models A \sqsubseteq B$ iff $\mathcal{M} \models A \sqsubseteq B$.*

2.2 Ontology Classification in HerMiT

The reasoner HerMiT implements a classification algorithm [6] that differs significantly from the standard Enhanced Traversal Algorithm [2] implemented in most other DL reasoners. The key feature of HerMiT's classification algorithm that makes it especially well-suited for our purposes is that it exploits sets \mathbf{K} and \mathbf{P} of pairs $\langle A, B \rangle$ of atomic concepts representing *known subsumptions* and *possible subsumptions*, respectively. These sets are used to reduce the number of required tests during classification. Information about non-subsumptions is implicitly stored in these sets (as it would be too costly to store it explicitly), i.e., if $\mathbf{A} = \{\langle A, B \rangle \mid A, B \text{ are atomic concept names in } \text{Sig}(\mathcal{O})\}$, then $\mathbf{A} \setminus (\mathbf{K} \cup \mathbf{P})$ is the set of known non-subsumptions.

The algorithm works in two clearly distinct phases. In the *initialisation phase*, sets \mathbf{K} and \mathbf{P} are given initial values using information obtained from satisfiability tests performed on atomic concepts. In the *classification phase*, \mathbf{K} is

augmented with pairs from \mathbf{P} until \mathbf{K} contains all the entailed subsumptions and \mathbf{P} is empty.

Additional technical details about HermiT’s classification algorithm are provided in the Appendix.

3 Modular Classification of Ontologies

Given a *SR \mathcal{OIQ}* ontology \mathcal{O} and a description logic $\mathcal{L} \subseteq \text{SR \mathcal{OIQ} }$, our first goal is to identify a signature $\Sigma^{\mathcal{L}} \subseteq \text{Sig}(\mathcal{O})$ such that $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} \subseteq \mathcal{O}_{\mathcal{L}}$. We call any such subset of $\text{Sig}(\mathcal{O})$ an *\mathcal{L} -signature for \mathcal{O}* . Section 3.1 addresses the problem of identifying as large an \mathcal{L} -signature as possible.

We can then use an \mathcal{L} -reasoner to compute from $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]}$ complete classification information about the atomic concepts in $\Sigma^{\mathcal{L}}$ —by Proposition 2, given any $A \in \Sigma^{\mathcal{L}}$ and $B \in \text{Sig}(\mathcal{O}) \cup \{\top, \perp\}$ we have $\mathcal{O} \models A \sqsubseteq B$ iff $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} \models A \sqsubseteq B$.

HermiT’s classification algorithm needs to be slightly modified in order to exploit the information computed by the \mathcal{L} -reasoner. In section 3.2 we show how to adapt the initialisation phase to efficiently encode this information into \mathbf{K} and \mathbf{P} . Additional technical information about our modification of HermiT’s algorithm (including a proof of correctness) is given in the Appendix.

3.1 Computing an \mathcal{L} -signature

The definition of \perp -module immediately suggests a simple “guess and check” algorithm for computing a (maximal) \mathcal{L} -signature for \mathcal{O} : consider all subsets $\Sigma \subseteq \text{Sig}(\mathcal{O})$ in decreasing size order and, for each of them, check whether $\mathcal{M}_{[\mathcal{O}, \Sigma]}$ is an \mathcal{L} -ontology.

Our goal in practice, however, is to optimise classification; hence, we propose a more practical algorithm. Although our algorithm is not guaranteed to compute a maximal \mathcal{L} -signature, it can be implemented very efficiently and, as shown in the evaluation section, it typically computes large \mathcal{L} -signatures, provided that $\mathcal{O}_{\mathcal{L}}$ is a large enough fragment of \mathcal{O} .

We will exploit the fact that every \mathcal{L} -signature $\Sigma^{\mathcal{L}}$ *must* satisfy the following property (\star) . If (\star) does not hold, then $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]}$ will contain some non \mathcal{L} -axiom.

$$\text{Property } (\star): \quad \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}} \text{ is } \perp\text{-local w.r.t. } \Sigma^{\mathcal{L}}$$

Example 1. Consider $\mathcal{L} = \mathcal{EL}$ and the following ontology

$$\mathcal{O}^{\text{ex}} = \{A \sqsubseteq B, \exists R.C \sqsubseteq D, E \sqsubseteq \forall S.A, \exists R.D \sqsubseteq \neg B\}$$

Note that the set of \mathcal{L} -axioms in \mathcal{O}^{ex} is $\mathcal{O}_{\mathcal{L}}^{\text{ex}} = \{A \sqsubseteq B, \exists R.C \sqsubseteq D\}$. Furthermore, the signature of $\mathcal{O}_{\mathcal{L}}^{\text{ex}}$, namely $\Sigma_1 = \{A, B, C, D, R\}$, is not an \mathcal{L} -signature for \mathcal{O}^{ex} ; indeed, the non \mathcal{L} -axiom $\exists R.D \sqsubseteq \neg B$ is not \perp -local w.r.t Σ_1 .

In contrast, we have that $\mathcal{O}^{\text{ex}} \setminus \mathcal{O}_{\mathcal{L}}^{\text{ex}} = \{E \sqsubseteq \forall S.A, \exists R.D \sqsubseteq \neg B\}$ is \perp -local w.r.t. $\Sigma_2 = (\text{Sig}(\mathcal{O}^{\text{ex}}) \setminus \text{Sig}(\mathcal{O}_{\mathcal{L}}^{\text{ex}})) = \{C\}$. Furthermore, $\mathcal{M}_{[\mathcal{O}^{\text{ex}}, \Sigma_2]} = \emptyset$; hence, Σ_2 is an \mathcal{L} -signature for \mathcal{O}^{ex} , and we can ensure that $\mathcal{O}^{\text{ex}} \not\models C \sqsubseteq X$ for each atomic concept $X \in \text{Sig}(\mathcal{O}^{\text{ex}})$ different from C . \diamond

Although Example 1 might suggest that property (\star) is also a *sufficient* condition for $\Sigma^{\mathcal{L}}$ to be an \mathcal{L} -signature in \mathcal{O} , this is unfortunately not the case.

Example 2. Consider $\Sigma_3 = \{A, C, D, R, S\}$; clearly, $\mathcal{O}^{\text{ex}} \setminus \mathcal{O}_{\mathcal{L}}^{\text{ex}}$ is \perp -local w.r.t Σ_3 and hence (\star) holds for Σ_3 . However, Σ_3 is not an \mathcal{L} -signature for \mathcal{O}^{ex} .

By Definition 4, each axiom in $\mathcal{O}^{\text{ex}} \setminus \mathcal{M}_{[\mathcal{O}^{\text{ex}}, \Sigma_3]}$ must be \perp -local w.r.t. signature $\Sigma_3 \cup \text{Sig}(\mathcal{M}_{[\mathcal{O}^{\text{ex}}, \Sigma_3]})$ (and not just w.r.t Σ_3). Axiom $\alpha = A \sqsubseteq B$ is not \perp -local w.r.t. Σ_3 , so we have $\alpha \in \mathcal{M}_{[\mathcal{O}^{\text{ex}}, \Sigma_3]}$. But then, we have $B \in \text{Sig}(\mathcal{M}_{[\mathcal{O}^{\text{ex}}, \Sigma_3]})$ and hence the non \mathcal{L} -axiom $\beta = \exists R.D \sqsubseteq \neg B$ is not \perp -local w.r.t. $\Sigma_3 \cup \text{Sig}(\mathcal{M}_{[\mathcal{O}^{\text{ex}}, \Sigma_3]})$.

We can address this problem by reducing Σ_3 to $\Sigma_4 = \Sigma_3 \setminus \{A\}$. The corresponding \perp -module for Σ_4 then becomes $\mathcal{M}_{[\mathcal{O}^{\text{ex}}, \Sigma_4]} = \{\exists R.C \sqsubseteq D\}$, which is an \mathcal{L} -ontology; thus, Σ_4 is an \mathcal{L} -signature for \mathcal{O}^{ex} . \diamond

Example 2 suggests an algorithm for computing an \mathcal{L} -signature for \mathcal{O} , which can be intuitively described as follows.

1. Reduce $\Sigma_0 = \text{Sig}(\mathcal{O})$ to a subset Σ_1 of Σ_0 such that $\mathcal{S}_0 = \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is \perp -local w.r.t. Σ_1 (thus satisfying (\star)).
2. Compute the axioms \mathcal{S}_1 in $\mathcal{M}_{[\mathcal{O}, \Sigma_1]}$ containing symbols not in Σ_1 .
3. Reduce Σ_1 to a subset Σ_2 of Σ_1 such that \mathcal{S}_1 is \perp -local w.r.t. Σ_2 .
4. Repeat Steps [2-4] until the set of axioms computed in Step 2 is empty.

Note that there can be many ways to perform the signature reduction required in Steps 1 and 4. For instance, Σ_2 and Σ_3 in Examples 1 and 2 are both possible reductions of $\text{Sig}(\mathcal{O}^{\text{ex}})$ in Step 1. These acceptable reductions can be characterised using a function

$$\text{localise} : \mathcal{P}(\text{Sig}(\mathcal{O})) \times \mathcal{P}(\mathcal{O}) \rightarrow \mathcal{P}(\text{Sig}(\mathcal{O}))$$

such that, given $\Sigma \in \mathcal{P}(\text{Sig}(\mathcal{O}))$ and $\mathcal{S} \in \mathcal{P}(\mathcal{O})$ not \perp -local w.r.t. Σ , $\text{localise}(\Sigma, \mathcal{S})$ returns

- Σ if $\mathcal{S} = \emptyset$.
- a subset $\Sigma' \subset \Sigma$ such that every axiom in \mathcal{S} is \perp -local w.r.t. Σ' if $\mathcal{S} \neq \emptyset$ and Σ' exists.
- \emptyset otherwise.

Given a particular `localise` function, Algorithm 1 accepts a *SRIOIQ* ontology \mathcal{O} and returns either the pair $\langle \text{false}, \emptyset \rangle$ or a pair $\langle \text{true}, \Sigma^{\mathcal{L}} \rangle$ with $\Sigma^{\mathcal{L}} \subseteq \text{Sig}(\mathcal{O})$ an \mathcal{L} -signature for \mathcal{O}^{ex} . Termination and correctness are granted by Theorem 1.

Theorem 1. *Let \mathcal{S}_i, Σ_i ($i \geq 0$) be defined by the following construction:*

$$\begin{aligned} (i = 0): \quad & \Sigma_0 = \text{Sig}(\mathcal{O}) & \mathcal{S}_0 = \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}} \\ (i \geq 1): \quad & \Sigma_i = \text{localise}(\Sigma_{i-1}, \mathcal{S}_{i-1}) & \mathcal{S}_i = \{\alpha \in \mathcal{M}_{[\mathcal{O}, \Sigma_i]} \mid \text{Sig}(\alpha) \not\subseteq \Sigma_i\} \end{aligned}$$

Let $\Sigma^{\mathcal{L}} := \bigcap_{i \geq 0} \Sigma_i$. Then, the following properties hold:

1. There exists $k < |\text{Sig}(\mathcal{O})|$ such that either $\Sigma_k = \emptyset$ or $\mathcal{S}_k = \emptyset$.
2. Either $\Sigma^{\mathcal{L}} = \emptyset$ or $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} \subseteq \mathcal{O}_{\mathcal{L}}$.

Algorithm 1 \mathcal{L} -signature(\mathcal{O})

Input: a \mathcal{SROIQ} ontology \mathcal{O}

```
1:  $\Sigma := \text{Sig}(\mathcal{O})$ 
2:  $\mathcal{S} := \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ 
3: canLocalise = true
4: while  $\mathcal{S} \neq \emptyset$  and canLocalise do
5:    $\Sigma := \text{localise}(\Sigma, \mathcal{S})$ 
6:   if  $\Sigma = \emptyset$  then
7:     canLocalise := false
8:   else
9:      $\mathcal{S} := \{\alpha \in \mathcal{M}_{[\mathcal{O}, \Sigma]} \mid \text{Sig}(\alpha) \not\subseteq \Sigma\}$ 
10: return  $\langle \text{canLocalise}, \Sigma \rangle$ 
```

Proof. We first show Claim 1. Suppose $\Sigma_i \neq \emptyset$ for each $i \geq 0$. A straightforward inductive argument would show that $\Sigma_j \subseteq \Sigma_i$ for each $j > i \geq 0$. Furthermore, $\Sigma_0 = \text{Sig}(\mathcal{O})$, so it cannot be the case that $\Sigma_j \subset \Sigma_i$ for each $0 \leq i < j \leq |\text{Sig}(\mathcal{O})|$. Therefore, there must be some $k < |\text{Sig}(\mathcal{O})|$ such that $\Sigma_{k+1} = \Sigma_k$; by the definition of `localise`, this implies that $\mathcal{S}_k = \emptyset$.

We finally show Claim 2. Suppose $\Sigma^{\mathcal{L}} \neq \emptyset$. It is enough to prove that each $\alpha \in \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is \perp -local w.r.t. $\Sigma^{\mathcal{L}} \cup \text{Sig}(\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}]})}$.

First, we are going to see that $\text{Sig}(\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}]})} \subseteq \Sigma^{\mathcal{L}}$. According to Claim 1, there exists $k < |\text{Sig}(\mathcal{O})|$ such that $\mathcal{S}_k = \emptyset$. This implies that, for each axiom $\alpha \in \mathcal{M}_{[\mathcal{O}, \Sigma_k]}$, we have $\text{Sig}(\alpha) \subseteq \Sigma_k$. It is easy to see that $\mathcal{S}_k = \emptyset$ also implies that $\Sigma_j = \Sigma_k$ for each $j > k$. Together with the fact that $\Sigma_j \subseteq \Sigma_i$ for each $j > i \geq 0$, this implies $\Sigma^{\mathcal{L}} = \bigcap_{i \geq 0} \Sigma_i = \Sigma_k$. But then for each $\alpha \in \mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} = \mathcal{M}_{[\mathcal{O}, \Sigma_k]}$ we have $\text{Sig}(\alpha) \subseteq \Sigma_k = \Sigma^{\mathcal{L}}$, and so $\text{Sig}(\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}]})} \subseteq \Sigma^{\mathcal{L}}$.

Now we can just prove that each $\alpha \in \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is \perp -local w.r.t. $\Sigma^{\mathcal{L}}$. Because $\Sigma^{\mathcal{L}} = \bigcap_{i \geq 0} \Sigma_i \neq \emptyset$, in particular it must be the case that $\Sigma_0 \neq \emptyset$. By definition of `localise`, either $\mathcal{O} \setminus \mathcal{O}_{\mathcal{L}} = \emptyset$ —in which case it is immediate that $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} \subseteq \mathcal{O}_{\mathcal{L}}$ —or every axiom in $\mathcal{S}_0 = \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is \perp -local w.r.t. $\Sigma_1 = \text{localise}(\Sigma_0, \mathcal{S}_0)$. Then, by Proposition 1, each $\alpha \in \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}}$ is \perp -local w.r.t. $\Sigma^{\mathcal{L}} \subseteq \Sigma_1$. \square

In practice, it is more convenient to use the \mathcal{L} -reasoner to classify $\mathcal{O}_{\mathcal{L}}$, instead of $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]}$. Once $\Sigma^{\mathcal{L}}$ has been computed, the following proposition shows that $\mathcal{O}_{\mathcal{L}}$ provides as much information as $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]}$ about the classification of \mathcal{O} . Furthermore, in general $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} \subset \mathcal{O}_{\mathcal{L}}$ so additional subsumption relationships might be obtained by classifying $\mathcal{O}_{\mathcal{L}}$.

Proposition 3. *Let $\Sigma^{\mathcal{L}}$ be an \mathcal{L} -signature for an ontology \mathcal{O} . Then for each atomic concept $A \in \Sigma^{\mathcal{L}}$ and each $B \in \text{Sig}(\mathcal{O}) \cup \{\top, \perp\}$ we have*

$$\mathcal{O} \models A \sqsubseteq B \text{ iff } \mathcal{O}_{\mathcal{L}} \models A \sqsubseteq B$$

Proof. Consider an atomic concept $A \in \Sigma^{\mathcal{L}}$ and $B \in \text{Sig}(\mathcal{O}) \cup \{\top, \perp\}$. By monotonicity, because $\mathcal{O}_{\mathcal{L}} \subseteq \mathcal{O}$, we know that

$$\mathcal{O} \not\models A \sqsubseteq B \text{ implies } \mathcal{O}_{\mathcal{L}} \not\models A \sqsubseteq B$$

Algorithm 2 \mathcal{L} -ModularClassification(\mathcal{O})

Input: a *SR \mathcal{O} I \mathcal{Q}* ontology \mathcal{O}

- 1: $\mathcal{O}_{\mathcal{L}} := \{\alpha \in \mathcal{O} \mid \alpha \text{ is an } \mathcal{L}\text{-axiom}\}$
 - 2: $\Sigma^{\mathcal{L}} := \mathcal{L}\text{-signature}(\mathcal{O})$ ▷ See Algorithm 1
 - 3: $H_{\mathcal{O}_{\mathcal{L}}} := \mathcal{L}\text{-classification}(\mathcal{O}_{\mathcal{L}})$
 - 4: $H := \text{HerMiTclassification}(\mathcal{O}_{\mathcal{L}}, H_{\mathcal{O}_{\mathcal{L}}}, \Sigma^{\mathcal{L}})$ ▷ See Section 3.2 and Appendix
 - 5: **return** H
-

By monotonicity, because $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} \subseteq \mathcal{O}_{\mathcal{L}}$ (by Theorem 1), it is the case that $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} \models A \sqsubseteq B$ implies $\mathcal{O}_{\mathcal{L}} \models A \sqsubseteq B$. Now $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]}$ is a \perp -module in \mathcal{O} w.r.t. $\Sigma^{\mathcal{L}}$, so by Proposition 2, $\mathcal{O} \models A \sqsubseteq B$ implies $\mathcal{M}_{[\mathcal{O}, \Sigma^{\mathcal{L}}]} \models A \sqsubseteq B$, and

$$\mathcal{O} \models A \sqsubseteq B \text{ implies } \mathcal{O}_{\mathcal{L}} \models A \sqsubseteq B$$

Therefore, for each atomic concept $A \in \Sigma^{\mathcal{L}}$ and $B \in \text{Sig}(\mathcal{O}) \cup \{\top, \perp\}$ we have $\mathcal{O} \models A \sqsubseteq B$ if and only if $\mathcal{O}_{\mathcal{L}} \models A \sqsubseteq B$, as required. \square

3.2 Adapting HerMiT's Initialisation Phase

As mentioned in Section 2.2, HerMiT's classification algorithm works with (disjoint) sets \mathbf{K} and \mathbf{P} of known and possible subsumptions, respectively. We next discuss how we can use the information extracted from $\mathcal{O}_{\mathcal{L}}$ by the \mathcal{L} -reasoner in the initialisation of \mathbf{K} and \mathbf{P} .

Let $\mathbf{K}' = \{\langle A, B \rangle \in \text{Sig}(\mathcal{O}) \times (\text{Sig}(\mathcal{O}) \cup \{\top, \perp\}) \mid \mathcal{O}_{\mathcal{L}} \models A \sqsubseteq B\}$ be the positive subsumptions extracted from $\mathcal{O}_{\mathcal{L}}$ by the \mathcal{L} -reasoner. We can clearly complement the initialisation of \mathbf{K} by simply adding \mathbf{K}' to \mathbf{K} .

To improve the initialisation of \mathbf{P} , we can simply make sure that no pair $\langle A, B \rangle \in \Sigma^{\mathcal{L}} \times \text{Sig}(\mathcal{O})$ is ever added to \mathbf{P} . Indeed, by Proposition 3, if $\mathcal{O} \models A \sqsubseteq B$ then $\langle A, B \rangle$ must already be in \mathbf{K}' ; otherwise, we must have $\mathcal{O} \not\models A \sqsubseteq B$ and there is no need to consider the pair $\langle A, B \rangle$ as a possible subsumption.

We include in the Appendix a slightly modified version of the initialisation algorithm in HerMiT that is capable of exploiting the information extracted from $\mathcal{O}_{\mathcal{L}}$ by the \mathcal{L} -reasoner in the way just explained.

Algorithm 2 describes, at an abstract level, how the entire classification process can be performed with our modular technique for a particular $\mathcal{L} \subseteq \text{SR \mathcal{O} I \mathcal{Q} }$ and a particular function localise.

4 Implementation and Experiments

We have implemented our algorithms in Java using the OWL API.² Our implementation of the localise function is based on the locality module extractor described in [12], which is publicly available.³

² <http://owlapi.sourceforge.net/>

³ <http://www.cs.ox.ac.uk/isg/tools/ModuleExtractor>

Table 1. Test ontologies

Ontology	Number of axioms		Signature	
	Total	\mathcal{EL}^{++}	Size	Concepts
SNOMED [⊔]	582,364	582,362	291,207	291,145
NCI	219,224	219,159	91,497	91,225
FMA-SNOMED	385,146	385,142	159,415	159,328

Table 2. \mathcal{L} -signature and classification times for $\mathcal{L} = \mathcal{EL}^{++}$

Ontology	$\Sigma^{\mathcal{L}}$			Classification time(s)	
	Size	Concepts	Time (s)	HermiT	Modular
SNOMED [⊔]	280,985 (96%)	280,923	15.3	2,016.5	189.9
NCI	85,411 (93%)	85,139	7.6	74.9	32.0
FMA-SNOMED	33,124 (21%)	33,046	14.3	876.5	790.6

In the implementation of `localise`, symbols required to make a set of axioms \perp -local are selected greedily axiom by axiom. When selecting symbols, we rely on heuristics that try to keep as many roles as possible within $\Sigma^{\mathcal{L}}$. This is because ontologies contain many more concepts than roles, and each role typically occurs in a large number of axioms; thus, having a role outside $\Sigma^{\mathcal{L}}$ is likely to cause many other symbols to be left outside $\Sigma^{\mathcal{L}}$.

In our experiments, we have used the ontologies given in Table 1:

- SNOMED[⊔] is a modification of the well-known SNOMED ontology (v. January 2010), where two axioms containing disjunction have been added (using feedback obtained from SNOMED’s developers).
- NCI is the latest version of the National Cancer Institute Thesaurus. This ontology contains 65 non \mathcal{EL}^{++} axioms.
- FMA-SNOMED is the ontology obtained from the integration of (a fragment of) the Foundational Model of Anatomy (FMA) and (a fragment of) SNOMED using ontology mappings [13]. In this case, all the non \mathcal{EL}^{++} axioms come from FMA.

Our results are summarised in Table 2. The first two columns in the table provide the total size and number of concepts in the \mathcal{EL}^{++} -signature. The third column indicates the time required to compute the \mathcal{EL}^{++} -signature using the algorithm described in Section 3.1. Finally, the last two columns provide the total classification time using (the latest version of) HermiT, and the classification time required to complete the classification of $\mathcal{O}_{\mathcal{L}}$ as described in Section 3.2. For convenience of implementation, we have also classified $\mathcal{O}_{\mathcal{L}}$ using HermiT (and this time has not been included in the table); however, the reasoner ELK can classify $\mathcal{O}_{\mathcal{L}}$ in all cases in just a few seconds (e.g., ELK can classify SNOMED using concurrent classification techniques in about 5 seconds [15]).

We can observe that 96% of the symbols in SNOMED[⊔] (and 93% of the symbols in NCI) are included in the \mathcal{EL}^{++} -signature; thus, all subsumers of concepts

in this signature can be completely determined using an \mathcal{EL}^{++} -reasoner. Note, however, that the size of the \mathcal{EL}^{++} -signature for FMA-SNOMED is comparatively much smaller. This is due to the structure of FMA, which contains several non \mathcal{EL}^{++} axioms about roles that are widely used in the ontology. For example, the domain of the role `hasMass` is defined as a disjunction of very general concepts, such as `MaterialThing`; since role `hasMass` is outside the \mathcal{EL}^{++} -signature, so will be `MaterialThing` (and, as a consequence, also the many concepts subsumed by `MaterialThing`).

Finally, concerning classification times, our results suggest the potential of our techniques. Improvements are especially substantial for both SNOMED^U and NCI, where the \mathcal{EL}^{++} -signature is very large.

5 Conclusion and Future Work

In this paper, we have proposed a technique for classifying a *SRIQ* ontology \mathcal{O} by exploiting a reasoner for a fragment \mathcal{L} of *SRIQ*. Our technique allows us to show that the subsumers of many concepts in \mathcal{O} can be completely determined using only the \mathcal{L} -reasoner. Although our implementation is still at a very prototypical stage, our preliminary experiments show the potential of our approach in practice.

Our work is only very preliminary, and there are many interesting possibilities for future work.

- Our heuristics for computing an \mathcal{L} -signature $\Sigma^{\mathcal{L}}$ are rather naive and there is plenty of room for improvement. For example, it might be possible to explore modular decomposition techniques to compute larger \mathcal{L} -signatures [22].
- Hermit’s initialisation phase could be further improved to make better use of the information obtained from the \mathcal{L} -reasoner.
- We are using \perp -modules, which provide very strong preservation guarantees (they preserve even *models*). It would be interesting to devise novel techniques for extracting modules that are more “permissive”, in the sense that they only provide preservation guarantees for atomic subsumptions.
- Our technique could also be applied to a different notion of locality, as long as it satisfied a result analogous to Proposition 2.
- It would be interesting to explore ontology rewriting techniques that complement module extraction. For example, we could rewrite \mathcal{O} into an \mathcal{L} -ontology \mathcal{O}' such that $\mathcal{O}' \models \mathcal{O}$, in which case the classification of \mathcal{O}' would provide an “upper bound” to the classification of \mathcal{O} .

Acknowledgements. This work was supported by the Royal Society, the EU FP7 project SEALS and the EPSRC projects ConDOR, ExODA, and LogMap.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: IJCAI (2005)

2. Baader, F., Franconi, E., Hollunder, B., Nebel, B., Profitlich, H.: An empirical analysis of optimization techniques for terminological representation systems. *Applied Intelligence* 4(2), 109–132 (1994)
3. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL - a polynomial-time reasoner for life science ontologies. In: *Proc. of IJCAR*. pp. 287–291 (2006)
4. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. *JAIR* 31, 273–318 (2008)
5. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. *J. Web Semantics (JWS)* 6(4), 309–322 (2008)
6. Glimm, B., Horrocks, I., Motik, B., Shearer, R., Stoilos, G.: A novel approach to ontology classification. *J. of Web Semantics* 10(1) (2011)
7. Grau, B.C., Halaschek-Wiener, C., Kazakov, Y., Suntisrivaraporn, B.: Incremental classification of description logics ontologies. *JAR* 44(4), 337–369 (2010)
8. Haarslev, V., Möller, R.: High performance reasoning with very large knowledge bases: A practical case study. In: *Proc. IJCAI*. pp. 161–168 (2001)
9. Haarslev, V., Möller, R.: Racer system description. In: *Proc. of IJCAR*. pp. 701–705 (2001)
10. Horrocks, I.: Implementation and optimisation techniques. In: *The Description Logic Handbook: Theory, Implementation, and Applications*, chap. 9, pp. 306–346 (2003)
11. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SROIQ*. In: *Proc. of KR*. pp. 57–67 (2006)
12. Jimenez-Ruiz, E., Cuenca Grau, B., Schneider, T., Sattler, U., Berlanga, R.: Safe and economic re-use of ontologies: a logic-based methodology and tool support. In: *Proc. of ESWC* (2008)
13. Jiménez-Ruiz, E., Grau, B.C.: Logmap: Logic-based and scalable ontology matching. In: *Proc. of ISWC* (2011)
14. Kazakov, Y.: *RIQ* and *SROIQ* are harder than *SHOIQ*. In: *Proc. of KR*. pp. 274–284 (2008)
15. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of \mathcal{EL} ontologies. In: *Proc. of ISWC*. vol. 7032 (2011)
16. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: *Proc. of IJCAI*. pp. 453–458 (2007)
17. Lutz, C., Wolter, F.: Conservative extensions in the lightweight description logic \mathcal{EL} . In: *Proc. of CADE-21*. vol. 4603 (2007)
18. Sattler, U., Schneider, T., Zakharyashev, M.: Which kind of module should I extract? In: *Proc. of DL* (2009)
19. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL DL reasoner. *J. of Web Semantics* 5(2), 51–53 (2007)
20. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: *Proc. of IJCAR*. vol. 4130, pp. 292–297 (2006)
21. Tsarkov, D., Horrocks, I., Patel-Schneider, P.: Optimizing terminological reasoning for expressive description logics. *JAR* 39(3), 277–316 (2007)
22. Vescovo, C.D., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition. In: *Proc. of IJCAI*. pp. 2232–2237 (2011)

UEL: Unification Solver for \mathcal{EL}

Franz Baader, Stefan Borgwardt, Julian Mendez, and Barbara Morawska*
{baader, stefborg, mendez, morawska}@tcs.inf.tu-dresden.de

Theoretical Computer Science, TU Dresden, Germany

Abstract. UEL is a system that computes unifiers for unification problems formulated in the description logic \mathcal{EL} . \mathcal{EL} is a description logic with restricted expressivity, but which is still expressive enough for the formal representation of biomedical ontologies, such as the large medical ontology SNOMED CT. We propose to use UEL as a tool to detect redundancies in such ontologies by computing unifiers of two formal concepts suspected of expressing the same concept of the application domain. UEL provides access to two different unification algorithms and can be used as a plug-in of the popular ontology editor Protégé, or stand-alone.

1 Motivation

The description logic (DL) \mathcal{EL} , which offers the concept constructors conjunction (\sqcap), existential restriction ($\exists r.C$), and the top concept (\top), has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in \mathcal{EL} [1,10,4]. On the other hand, though quite inexpressive, \mathcal{EL} can be used to define biomedical ontologies, such as the large medical ontology SNOMED CT.¹

Unification in DLs has been proposed in [9] as a novel inference service that can, for instance, be used to detect redundancies in ontologies. For example, assume that one developer of a medical ontology defines the concept of a *patient with severe head injury* as

$$\text{Patient} \sqcap \exists \text{finding} . (\text{Head_injury} \sqcap \exists \text{severity} . \text{Severe}), \quad (1)$$

whereas another one represents it as

$$\text{Patient} \sqcap \exists \text{finding} . (\text{Severe_injury} \sqcap \exists \text{finding_site} . \text{Head}). \quad (2)$$

These two concept descriptions are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by treating the concept names `Head_injury` and `Severe_injury` as variables, and substituting the first one by `Injury` \sqcap $\exists \text{finding_site} . \text{Head}$ and the second one by `Injury` \sqcap $\exists \text{severity} . \text{Severe}$. In this case, we say that the descriptions are unifiable, and call the substitution that makes them equivalent a *unifier*. Intuitively, such

* Supported by DFG under grant BA 1122/14-1

¹ see <http://www.ihtsdo.org/snomed-ct/>

Table 1. Syntax and semantics of \mathcal{EL}

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top	\top	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$

a unifier proposes definitions for the concept names that are used as variables: in our example, we know that, if we define `Head_injury` as `Injury` \sqcap \exists `finding_site`.`Head` and `Severe_injury` as `Injury` \sqcap \exists `severity`.`Severe`, then the two concept descriptions (1) and (2) are equivalent w.r.t. these definitions. Of course, this example was constructed such that the unifier actually provides sensible definitions for the concept names used as variables. In general, the existence of a unifier only says that there is a structural similarity between the two concepts. The developer that uses unification as a tool for finding redundancies in an ontology or between two different ontologies needs to inspect the unifier(s) to see whether the suggested definitions really make sense.

In [6] it was shown that unification in \mathcal{EL} is an NP-complete problem. Basically, this problem is in NP since every solvable unification problem has a “local” unifier, i.e., one built from parts of the unification problem. The NP algorithm introduced in [6] is a brutal “guess and then test” algorithm, which guesses a local substitution and then checks whether it is a unifier. In [8], a more practical rule-based \mathcal{EL} -unification algorithm was introduced, which tries to transform the given unification problems into a solved form, and makes nondeterministic decisions only if triggered by the problem. Finally, the paper [7] proposes a third algorithm, which encodes the unification problem into a set of propositional clauses and then solves it using an existing highly optimized SAT solver.

Version 1.0.0 of our system UEL² used only the SAT translation to solve unification problems. This approach allowed us to get a fast unification algorithm—the unification problem only has to be translated into a propositional formula and the SAT solver is used to actually solve the problem. In contrast, for the implementation of the rule-based algorithm from [8] we had to find efficient methods to deal with the nondeterminism ourselves. As of version 1.2.0, UEL includes both implementations, as well as a variant of the SAT translation that tries to compute only small unifiers.

2 \mathcal{EL} and Unification in \mathcal{EL}

In order to explain what the algorithms implemented in UEL actually compute, we need to recall the relevant definitions and results for unification in \mathcal{EL} .

² All versions of this system are available at <http://uel.sourceforge.net>.

Starting with a finite set N_C of *concept names* and a finite set N_R of *role names*, \mathcal{EL} -*concept descriptions* are built from concept names using the constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$ for every $r \in N_R$), and *top* (\top). On the semantic side, concept descriptions are interpreted as sets. To be more precise, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. This function is extended to concept descriptions as shown in the semantics column of Table 1.

A *concept definition* is of the form $A \equiv C$ for a concept name A and a concept description C . A *TBox* \mathcal{T} is a finite set of concept definitions such that no concept name occurs more than once on the left-hand side of a definition in \mathcal{T} . The TBox \mathcal{T} is called *acyclic* if there are no cyclic dependencies between its concept definitions. Given a TBox \mathcal{T} , we call a concept name A a *defined concept* if it occurs as the left-side of a concept definition $A \equiv C$ in \mathcal{T} . All other concept names are called *primitive concepts*. An interpretation \mathcal{I} is a *model* of a TBox \mathcal{T} if $A^{\mathcal{I}} = C^{\mathcal{I}}$ holds for all definitions $A \equiv C$ in \mathcal{T} .

Subsumption asks whether a given concept description C is a subconcept of another concept description D : C is *subsumed* by D w.r.t. \mathcal{T} ($C \sqsubseteq_{\mathcal{T}} D$) if every model \mathcal{I} of \mathcal{T} satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We say that C is *equivalent* to D w.r.t. \mathcal{T} ($C \equiv_{\mathcal{T}} D$) if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. For the empty TBox, we write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\emptyset} D$ and $C \equiv_{\emptyset} D$, and simply talk about subsumption and equivalence (without saying “w.r.t. \emptyset ”).

In order to define unification, we partition the set N_C of concept names into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). Intuitively, N_v are the concept names that have possibly been given another name or been specified in more detail in another concept description describing the same notion. A *substitution* σ maps every variable to a concept description. It can be extended to concept descriptions in the usual way.

Unification in \mathcal{EL} was first considered w.r.t. the empty TBox [6]. In this setting, an \mathcal{EL} -*unification problem* is a finite set $\Gamma = \{C_1 \equiv? D_1, \dots, C_n \equiv? D_n\}$ of equations. A substitution σ is a *unifier* of Γ if σ *solves* all the equations in Γ , i.e., if $\sigma(C_1) \equiv \sigma(D_1), \dots, \sigma(C_n) \equiv \sigma(D_n)$. We say that Γ is *solvable* if it has a unifier. Without loss of generality, we can assume that the unification problem is *flat*, i.e., that all concept descriptions occurring in it are conjunctions of concept names or existential restrictions of the form $\exists r.A$ with $A \in N_C$. If a unification problem is not flat, we can transform it into a flat unification problem by introducing auxiliary variables.

As mentioned before, the main reason for solvability of unification in \mathcal{EL} to be in NP is that any solvable unification problem has a local unifier. Basically, any unification problem Γ determines a polynomial number of so-called *non-variable atoms*, which are concept constants or existential restrictions of the form $\exists r.A$ for a role name r and a concept constant or variable A . An *assignment* S maps every concept variable X to a subset S_X of the set At_{nv} of non-variable atoms of Γ . Such an assignment induces a relation $>_S$ on N_v , which is the transitive

closure of $\{(X, Y) \in N_v \times N_v \mid Y \text{ occurs in an element of } S_X\}$. We call the assignment S *acyclic* if $>_S$ is irreflexive (and thus a strict partial order). Any acyclic assignment S induces a unique substitution σ_S , which can be defined by induction along $>_S$:

- If $X \in N_v$ is minimal w.r.t. $>_S$, then we define $\sigma_S(X) := \prod_{D \in S_X} D$.
- Assume that $\sigma_S(Y)$ is already defined for all Y such that $X >_S Y$. Then we define $\sigma_S(X) := \prod_{D \in S_X} \sigma_S(D)$.

We call a substitution σ *local* if it is of this form, i.e., if there is an acyclic assignment S such that $\sigma = \sigma_S$. Consequently, one can enumerate (or guess, in a nondeterministic machine) all acyclic assignments and then check whether any of them induces a substitution that is a unifier. Using this brute-force approach, in general many local substitutions will be generated that only in the subsequent check turn out not to be unifiers.

The fact that any solvable unification problem has a local unifier was shown in [6] by proving that such a problem always has a *minimal* unifier and that every minimal unifier is equivalent to a local unifier. Minimality and equivalence of unifiers are determined by the following order \succeq on substitutions. For two substitutions σ and θ , we define $\sigma \succeq \theta$ iff $\sigma(X) \sqsubseteq \theta(X)$ holds for all variables X . We say that a unifier σ of a unification problem Γ is *minimal* if there is no unifier γ of Γ such that $\sigma \succeq \gamma$ and $\gamma \not\preceq \sigma$. Two unifiers σ, θ are *equivalent* iff $\sigma \succeq \gamma$ and $\gamma \succeq \sigma$. We introduce a similar order \succeq on assignments and write $S \succeq S'$ iff $S_X \supseteq S'_X$ holds for all $X \in N_v$. We call an assignment S *minimal* if σ_S is a unifier of Γ and there is no assignment S' different from S such that $\sigma_{S'}$ is a unifier of Γ and $S \succeq S'$. The following is easy to show: if $S \succeq S'$, then $\sigma_S \succeq \sigma_{S'}$. As shown in [3], this implies that all minimal unifiers are induced by minimal assignments, but the opposite need not hold. Computing only the minimal assignments is thus a first step towards computing only minimal unifiers. Computing only minimal unifiers is desirable since they are those among the local unifiers that substitute the variables by smaller concept descriptions and their corresponding assignments do not contain “irrelevant” non-variable atoms.

In [8], *unification w.r.t. an acyclic TBox* \mathcal{T} was introduced. In this setting, the concept variables are a subset of the primitive concepts of \mathcal{T} , and substitutions are applied both to the concept descriptions in the unification problem and to the right-hand sides of the definitions in \mathcal{T} . To deal with such unification problems, one does not need to develop a new algorithm. In fact, by viewing the defined concepts of \mathcal{T} as variables, one can turn \mathcal{T} into a unification problem, which one simply adds to the given unification problem Γ . As shown in [8], there is a 1–1-correspondence between the unifiers of Γ w.r.t. \mathcal{T} and the unifiers of this extended unification problem.

We will now describe the two unification algorithms implemented in UEL 1.2.0. Both algorithms have in common that they generate acyclic assignments, and thus output only local unifiers. They follow two different approaches to reduce the amount of blind guessing of the brute-force approach.

The SAT Translation

Instead of blindly generating all local substitutions, the reduction to the propositional satisfiability problem introduced in [7] ensures that only assignments that induce unifiers are generated. The set of propositional clauses $C(\Gamma)$ generated by the reduction contains two kinds of propositional letters: $[A \sqsubseteq B]$ for $A, B \in \text{At}_{\text{nv}}$ ³ and $[X > Y]$ for concept variables X, Y . Intuitively, setting $[A \sqsubseteq B] = 1$ means that the local substitution σ_S induced by the corresponding assignment S satisfies $\sigma_S(A) \sqsubseteq \sigma_S(B)$, and setting $[X > Y] = 1$ means that $X >_S Y$. The clauses in $C(\Gamma)$ are such that Γ has a unifier iff $C(\Gamma)$ is satisfiable. In particular, any propositional valuation τ satisfying $C(\Gamma)$ defines an assignment S^τ with $S_X^\tau := \{A \mid \tau([X \sqsubseteq A]) = 1, A \in \text{At}_{\text{nv}}\}$, which induces a local unifier of Γ . Conversely, any local unifier of Γ can be obtained in this way. Thus, by generating all propositional valuations satisfying $C(\Gamma)$ we can generate all local unifiers of Γ . The main advantage of this algorithm is the speed of the used SAT solver. However, the number of generated clauses is in general cubic in the size of the unification problem. Thus, the translation will generate huge SAT instances even from moderately sized unification problems, which might lead to memory problems even before the SAT solver is applied.

The Rule-Based Algorithm

The second unification algorithm implemented in UEL is based on the rule-based algorithm from [8]. However, internally it uses subsumptions of the form $C \sqsubseteq^? D$ instead of equivalences. This is without loss of generality since any equivalence $C \equiv^? D$ can be expressed by the two subsumptions $C \sqsubseteq^? D$ and $D \sqsubseteq^? C$. This variant of the algorithm has been described in more detail in [2].

The algorithm generates local unifiers by maintaining a set of current subsumptions Γ and a current acyclic assignment S , both of which are extended by applying certain rules. Initially, all subsumptions are marked as *unsolved* and rules apply only to unsolved subsumptions and mark them as *solved*. In the process, new subsumptions may be generated and the current assignment may be extended by adding non-variable atoms to some of the sets S_X . Once all subsumptions are solved, the substitution σ_S induced by the current assignment is a unifier of the unification problem.

Some of the rules are don't-know nondeterministic, i.e., they might apply in different ways to the same subsumption, but we do not know beforehand which application is the correct one. Also the choice between several applicable nondeterministic rules is don't-know nondeterministic. The algorithm additionally employs several *eager* rules that are always applied first and leave no choice in their application. They are mainly there to reduce the number of nondeterministic choices the algorithm has to make.

In contrast to the SAT reduction, this rule-based algorithm does not generate all local unifiers. A non-variable atom D will only be put in the set S_X if there

³ The reduction in [7] actually uses variables $[A \sqsubseteq B]$, but it turned out that the reduction using non-negated subsumptions behaves better in practice.

is a reason to do so in the unification problem, although there may be a local unifier whose assignment S' contains D in S'_X . However, it was shown in [8] that it can generate all minimal unifiers (up to equivalence). The converse is not true, i.e., it might also generate local unifiers that are not minimal.

The main advantage of this algorithm is that non-variable atoms are only added to the assignment if this is required by the unification problem, and thus fewer unifiers of relatively small size are generated. The space requirements are also quite low, since the current set of subsumptions and the current assignment are of size at most quadratic in the input size. The downside of this algorithm is that, without any of the optimizations implemented in modern SAT solvers, the algorithm naively traverses the search space. However, implementing such optimizations to improve its efficiency requires a huge effort.

3 Stuff not Mentioned in the Theoretical Papers

When implementing UEL, we had to deal with several issues that are abstracted away in the theoretical papers describing unification algorithms for \mathcal{EL} . Most of them are not specific to the used unification algorithm.

Primitive definitions In addition to concept definitions, as introduced above, biomedical ontologies often contain so-called *primitive definitions* $A \sqsubseteq C$ where A is a concept name and C is a concept description. Models \mathcal{I} of $A \sqsubseteq C$ need to satisfy $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. Thus, primitive definitions formulate necessary conditions for concept membership, but these conditions are not sufficient. SNOMED CT contains about 350,000 primitive definitions and only 40,000 concept definitions.

By using a trick first introduced by Nebel [13], primitive definitions $A \sqsubseteq C$ can be turned into concept definitions $A \equiv C \sqcap A_UNDEF$, where A_UNDEF is a new concept name that stands for the undefined part of the definition of A . In the resulting acyclic TBox, these new concept names are primitive concepts, and thus can be declared to be variables. In this case, a unifier σ suggest how to complete the definition of A by providing the concept description $\sigma(A_UNDEF)$.

Unifiers as acyclic TBoxes Given an acyclic assignment S computed by one of the unification algorithms, our system UEL actually does not produce the corresponding local unifier σ_S as output, but rather the acyclic TBox $\mathcal{T}_S := \{X \equiv \prod_{D \in S_X} D \mid X \in N_v\}$. This TBox solves the input unification problem Γ w.r.t. \mathcal{T} in the sense that $C \equiv_{\mathcal{T} \cup \mathcal{T}_S} D$ holds for all equations $C \equiv? D$ in Γ . This is actually what the developer that employs unification wants to know: how must the concept variables be defined such that the concept descriptions in the equations become equivalent? Another advantage of this representation of the output is that the size of S and thus of \mathcal{T}_S is polynomial in the size of the input Γ and \mathcal{T} , while the size of the concept descriptions $\sigma_S(X)$ may be exponential in this size. In the following, we will also call the TBoxes \mathcal{T}_S unifiers.

Internal variables As mentioned before, the unification algorithms for \mathcal{EL} assume that the unification problem is first transformed into a flat form. This form

can easily be generated by introducing auxiliary variables. These new variables have system-generated names, which do not make sense to the user. Thus, they should not show up in the output acyclic TBox \mathcal{T}_S . By replacing such auxiliary defined concepts in \mathcal{T}_S by their definitions as long as auxiliary names occur, we can transform \mathcal{T}_S into an acyclic TBox that satisfies this requirement, actually without causing an exponential blow-up of the size of the TBox.

Reachable subontology As mentioned above, acyclic TBoxes are treated by viewing them as part of the unification problem. For very large TBoxes like SNOMED CT, adding the whole TBox to the unification problem is neither viable nor necessary. In fact, it is sufficient to add the reachable part of the TBox, i.e., the definitions on which the concept descriptions in the unification problem depend. This reachable part is usually rather small, even for very large ontologies.

Enumeration of unifiers While computing a single unifier is usually quite fast, computing all of them can take much longer. We alleviate this problem by enabling the user to compute and then inspect one unifier at a time. If this unifier makes sense, i.e., suggests reasonable definitions for the variables, then the user can stop. Otherwise, the computation of the next unifier can be initiated.

For the rule-based algorithm, we output all produced unifiers by a depth-first traversal of the search space. This means that we apply applicable nondeterministic rules as long as this is possible. If there are no more unsolved subsumptions, we return the corresponding unifier. If no rule is applicable, we backtrack and apply the last nondeterministic rule in a different way or apply another rule.

For the SAT reduction, the approach is different. If the SAT solver has provided a satisfying propositional valuation, we can add a clause to the SAT problem that prevents the re-computation of this assignment, and call the SAT solver with this new SAT instance. If the SAT solver determines that the current set of clauses is unsatisfiable, then there are no more unifiers.

Computing only minimal assignments The satisfying valuations of the propositional formula generated by the SAT translation yield *all* local unifiers of the unification problem. Depending on how many concept names are turned into variables, there can be many local unifiers. To address this problem, we implemented a variant of the SAT reduction that computes only the local unifiers induced by *minimal* assignments w.r.t. \succ , which are often significantly fewer.

This approach is still complete in the sense that it computes all minimal unifiers (see Section 2). It works by transforming the SAT problem into a special case of a partial MAX-SAT problem [11], where in addition to the clauses that are to be satisfied one can specify a subset V_{\min} of the propositional variables. The goal is to minimize the number of variables from the set V_{\min} that are set to 1. By setting $V_{\min} = \{[X \sqsubseteq A] \mid X \in N_v, A \in \text{At}_{nv}\}$, we ensure that the first valuation returned by the MAX-SAT solver induces a minimal assignment S .⁴

⁴ If $\{(X, A) \mid X \in N_v, A \in S_X\}$ has minimal cardinality among all assignments that induce a unifier of the unification problem, then it is also minimal w.r.t. set inclusion.

To ensure that subsequent calls to the MAX-SAT solver return no assignments larger than S , we add a clause to the problem instance that requires that at least one of the variables of the form $[X \sqsubseteq A]$ with $A \in S_X$ should be set to 0.

Of course, the reformulation of the problem as a MAX-SAT instance adds an overhead to the computation time. However, this approach guarantees that no “superfluous,” i.e., non-minimal, assignments are presented to the user.

4 The User Interface

UEL was implemented in Java 1.6 and is compatible with Java 1.7. It uses the OWL API 3.2.4⁵ to read ontologies. It has a visual interface that can be used as a Protégé 4.1 plug-in, or as a standalone application. For the SAT-based unification algorithm, we currently use SAT4J⁶ as (MAX-)SAT solver, which is implemented in Java. However, this configuration can easily be changed to any solver that accepts the popular DIMACS CNF⁷ (or WCNF⁸) format as input and returns the computed satisfying propositional valuation. For the rule-based algorithm, we have implemented everything from scratch, and thus have no external dependencies.

After opening UEL’s visual interface, the first step is to open one or two ontologies. The latter enables unification of concepts defined in different ontologies. Additionally, the user can choose between the three unification algorithms by selecting the “SAT-based algorithm [(minimal assignments)]” or the “Rule-based algorithm”. The user can then choose two concepts to be unified. This is done by choosing two concept names that occur on the left-hand sides of concept definitions or primitive definitions (see Figure 1). UEL then computes the subontologies reachable from these concept names, and turns the primitive definitions in these subontologies into concept definitions.

After choosing the concepts to be unified, pressing the button  opens a dialog window in which the user is presented with the primitive concepts contained in these subontologies (including the ones with ending `_UNDEF`). The user can then decide which of these primitive concepts should be viewed as variables in the unification problem.

Once the user has chosen the variables, UEL computes the unification problem defined this way and opens a dialog window with control buttons. By pressing the button , the user triggers the computation of the first (or next) unifier. Each computed unifier is shown as an acyclic TBox in KRSS format. The button  can be used to go back to the previously computed unifier. The button  can be used to trigger the computation of all remaining unifiers, and the button  allows to jump back to the first unifier (see Figure 2). Computed unifiers are stored, and thus need not be recomputed during navigation. Each unifier (i.e., the acyclic TBox representing it) can be saved using the RDF/OWL or the

⁵ <http://owlapi.sourceforge.net>

⁶ <http://www.sat4j.org>

⁷ <http://www.satcompetition.org/2011/format-benchmarks2011.html>

⁸ <http://www.maxsat.udl.cat/11/requirements/index.html>

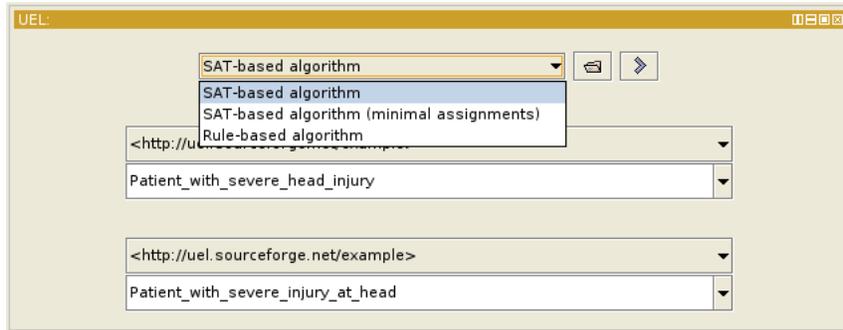


Fig. 1. The user can choose the unification algorithm and the concepts to be unified.



Fig. 2. The user can browse through the computed unifiers.

KRSS format by pressing the button . The file format is determined by the filename extension given by the user (.owl or .krss).

The user can use the button to retrieve internal details about the computation process. The unification problem created internally by UEL is then shown in KRSS format in a separate dialog. Additionally, the number of all concept variables (those chosen by the user and internal variables) is given. Depending on the chosen algorithm, several other internal statistics can be viewed. If the SAT reduction is used, the number of propositional letters and the number of propositional clauses are listed. For the rule-based algorithm, the number of initial subsumptions, the maximal number of generated subsumptions (over all nondeterministic choices), and the size of the search tree (i.e., the number of nondeterministic choices) are shown, as well as the number of “dead ends” where the algorithm had to backtrack. These numbers always reflect the current status and might increase if more unifiers are computed.

5 Some Examples

To illustrate the behavior of the three approaches, we consider several example problems. The size of these problems as well as the size of the generated data structures (propositional clauses or subsumptions) is depicted in Table 2, together with the runtimes of the three algorithms on these problems.

The first example is a modified version of our example from the beginning, where the TBox gives (1) as definition for `Patient_with_severe_head_injury` and (2) as definition for `Patient_with_severe_injury_at_head`. In addition, the TBox contains two primitive definitions, saying that `Head_injury` and `Severe_injury`

Table 2. These are the numbers of equations and variables of some example problems, as well as the numbers of generated clauses, propositional variables, and subsumptions, and the number of dead ends encountered by the rule-based algorithm. The second part contains the runtime and number of computed unifiers of each algorithm.

#	problem size						SAT		MAX-SAT		Rule	
	equ.	var.	clauses	prop.	subs. (max.)	dead ends	time	unif.	time	unif.	time	unif.
1	7	8	3,976	320	20 (52)	0	0.249	128	0.047	1	0.001	1
2	23	12	17,971	820	47 (217)	63,523	0.500	0	0.510	0	1.920	0
3	34	14	28,071	1,096	62 (320)	1,126,286	1.086	15	1.162	15	26.308	30

are subconcepts of Injury. If we choose Patient_with_severe_head_injury and Patient_with_severe_injury_at_head as the concepts to be unified, the system offers us the primitive concepts Patient, Severe, Head, Head_injury_UNDEF, and Severe_injury_UNDEF as possible variables, of which we choose only the latter two.

The SAT translation generates a large SAT problem (4,000 clauses are created from only 7 equations) and first computes the following unifier:

$$\{\text{Head_injury_UNDEF} \mapsto \exists \text{finding_site.Head}, \\ \text{Severe_injury_UNDEF} \mapsto \exists \text{severity.Severe}\}.$$

This substitution completes the primitive definitions of the concepts Head_injury and Severe_injury to concept definitions $\text{Head_injury} \equiv \text{Injury} \sqcap \text{finding_site.Head}$ and $\text{Severe_injury} \equiv \text{Injury} \sqcap \exists \text{severity.Severe}$.

However, the unification problem has 127 additional local unifiers. Some of them are similar to the first one, but contain “redundant” conjuncts. Others do not make much sense in the application (e.g., ones where Patient occurs in the images of the variables). In contrast, the MAX-SAT variant of the translation has to deal with an even larger problem since it has to consider the additional set V_{\min} . However, it is much faster since it computes only the unifier shown above, which is the only minimal unifier of this unification problem. The rule-based algorithm shows an even better performance since the data structures it creates are orders of magnitude smaller than the SAT problem and the unification problem is very deterministic—in fact, no backtracking is necessary. It also returns only the minimal unifier.

The second unification problem was constructed starting with a “hard” SAT instance (according to the approach in [12]), which was translated into a unification problem using the construction in [5]. Even though a large number of propositional clauses are constructed in the reduction back to a SAT problem, the SAT solver takes little time to determine that the problem has no solution. The overhead incurred by the use of a MAX-SAT problem is negligible. In contrast, the rule-based algorithm works on a much smaller data structure (at most 217 subsumptions during the whole run), but lacks the optimizations of the SAT solver and naively traverses a large search tree looking for a unifier.

The last example shows even more extreme differences. Again, we started from a simple SAT problem (“Choose exactly 2 out of 6 variables.”) that has 15

models which correspond to 15 minimal unifiers. Both the SAT-based and the MAX-SAT-based approach compute these unifiers quite fast, but the rule-based algorithm takes much longer and even computes each of the solutions twice.

6 Conclusions

Our system UEL enables ontology engineers to test ontologies for redundancies and allows them to choose between different algorithms with different strengths. The rule-based algorithm has a big advantage over the SAT translation since it needs only small data structures, but currently lacks important search optimizations, which make this implementation unsuitable for large problems.

In the current version 1.2.0, UEL only supports checking two pre-selected concept names for similarities. In future versions, we plan to implement an automatic search feature that can scan (a part of) an ontology for concept names that unify. For this we also need to find an intelligent way to automatically select the variables from a set of primitive concept names.

References

1. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proc. IJCAI'03. pp. 325–330. Morgan Kaufmann (2003)
2. Baader, F., Borgwardt, S., Morawska, B.: Unification in the description logic \mathcal{EL} w.r.t. cycle-restricted TBoxes. LTCS-Report 11-05, Chair of Automata Theory, TU Dresden, Germany (2011), see <http://lat.inf.tu-dresden.de/research/reports.html>.
3. Baader, F., Borgwardt, S., Morawska, B.: Computing minimal \mathcal{EL} -unifiers is hard. LTCS-Report 12-03, Chair for Automata Theory, TU Dresden (2012), see <http://lat.inf.tu-dresden.de/research/reports.html>.
4. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Proc. IJCAI'05. pp. 364–369. Professional Book Center (2005)
5. Baader, F., Küsters, R.: Matching concept descriptions with existential restrictions. In: Proc. KR'00. pp. 261–272. Morgan Kaufmann (2000)
6. Baader, F., Morawska, B.: Unification in the description logic \mathcal{EL} . In: Proc. RTA'09. LNCS, vol. 5595, pp. 350–364. Springer (2009)
7. Baader, F., Morawska, B.: SAT encoding of unification in \mathcal{EL} . In: Proc. LPAR'10. LNCS, vol. 6397, pp. 97–111. Springer (2010)
8. Baader, F., Morawska, B.: Unification in the description logic \mathcal{EL} . Log. Meth. Comput. Sci. 6(3) (2010)
9. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. Symb. Comput. 31(3), 277–305 (2001)
10. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: Proc. ECAI'04. pp. 298–302 (2004)
11. Li, C.M., Manyà, F.: Maxsat, hard and soft constraints. In: Handbook of Satisfiability, chap. 19, pp. 613–631. IOS Press (2009)
12. Markström, K.: Locality and hard SAT-instances. JSAT 2(1-4), 221–227 (2006)
13. Nebel, B.: Reasoning and Revision in Hybrid Representation Systems, LNAI, vol. 422. Springer (1990)

A Goal-Oriented Algorithm for Unification in \mathcal{EL} w.r.t. Cycle-Restricted TBoxes*

Franz Baader, Stefan Borgwardt, and Barbara Morawska
{baader, stefborg, morawska}@tcs.inf.tu-dresden.de

Theoretical Computer Science, TU Dresden, Germany

1 Introduction

Unification in DLs has been proposed in [7] (for the DL \mathcal{FL}_0 , which offers the constructors conjunction (\sqcap), value restriction ($\forall r.C$), and the top concept (\top)) as a novel inference service that can, for instance, be used to detect redundancies in ontologies. For example, assume that one developer of a medical ontology defines the concept of a *patient with severe head injury* as

$$\text{Patient} \sqcap \exists \text{finding} . (\text{Head_injury} \sqcap \exists \text{severity} . \text{Severe}), \quad (1)$$

whereas another one represents it as

$$\text{Patient} \sqcap \exists \text{finding} . (\text{Severe_finding} \sqcap \text{Injury} \sqcap \exists \text{finding_site} . \text{Head}). \quad (2)$$

These two concept descriptions are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by treating the concept names `Head_injury` and `Severe_finding` as variables, and substituting the first one by `Injury` \sqcap `finding_site.Head` and the second one by `severity.Severe`. In this case, we say that the descriptions are unifiable, and call the substitution that makes them equivalent a *unifier*. Intuitively, such a unifier proposes definitions for the concept names that are used as variables: in our example, we know that, if we define `Head_injury` as `Injury` \sqcap `finding_site.Head` and `Severe_finding` as `severity.Severe`, then the two concept descriptions (1) and (2) are equivalent w.r.t. these definitions. Here equivalence holds without additional GCIs.

To motivate our interest in unification w.r.t. GCIs, assume that the second developer uses the description

$$\begin{aligned} & \text{Patient} \sqcap \exists \text{status} . \text{Emergency} \sqcap \\ & \exists \text{finding} . (\text{Severe_finding} \sqcap \text{Injury} \sqcap \exists \text{finding_site} . \text{Head}) \end{aligned} \quad (3)$$

instead of (2). The descriptions (1) and (3) are not unifiable without additional GCIs, but they are unifiable, with the same unifier as above, if the GCI

$$\exists \text{finding} . \exists \text{severity} . \text{Severe} \sqsubseteq \exists \text{status} . \text{Emergency}$$

* Supported by DFG under grant BA 1122/14-1

is present in a background ontology.

In [4], we were able to show that unification in the DL \mathcal{EL} (which differs from \mathcal{FL}_0 by offering existential restrictions $(\exists r.C)$ in place of value restrictions) is of considerably lower complexity than in \mathcal{FL}_0 : the decision problem in \mathcal{EL} is NP-complete rather than EXPTIME-complete in \mathcal{FL}_0 . In addition to a brute-force “guess and then test” NP-algorithm [4], we have developed a goal-oriented unification algorithm for \mathcal{EL} , in which nondeterministic decisions are only made if they are triggered by “unsolved parts” of the unification problem [6], and an algorithm that is based on a reduction to satisfiability in propositional logic (SAT) [5], which enables the use of highly-optimized SAT solvers. In [6] it was also shown that the approaches for unification of \mathcal{EL} -concept descriptions (without any background ontology) can easily be extended to the case of an acyclic TBox as background ontology without really changing the algorithms or increasing their complexity. Basically, by viewing defined concepts as variables, an acyclic TBox can be turned into a unification problem that has as its unique unifier the substitution that replaces the defined concepts by unfolded versions of their definitions. For GCIs, this simple trick is not possible.

In [2], we extended the brute-force “guess and then test” NP-algorithm from [4] to the case of GCIs, which required the development of a new characterization of subsumption w.r.t. GCIs in \mathcal{EL} . Unfortunately, the algorithm is complete only for general TBoxes (i.e., finite sets of GCIs) that satisfy a certain restriction on cycles, which, however, does not prevent all cycles. For example, the cyclic GCI $\exists \text{child.Human} \sqsubseteq \text{Human}$ satisfies this restriction, whereas the cyclic GCI $\text{Human} \sqsubseteq \exists \text{parent.Human}$ does not.

In the present paper, we describe a goal-oriented algorithm for unification in \mathcal{EL} w.r.t. cycle-restricted general TBoxes, which extends the one from [6] and reduces the amount of nondeterministic guesses considerably. Full proofs of the presented results can be found in [1].

2 The Description Logic \mathcal{EL}

Syntax and semantics of \mathcal{EL} are defined in the usual way (see, e.g., [9]). Here, we just recall that \mathcal{EL} -*concept descriptions* are built from a finite set N_C of *concept names* and a finite set N_R of *role names* using the concept constructors *top-concept* (\top), *conjunction* ($C \sqcap D$), and *existential restriction* ($\exists r.C$ for every $r \in N_R$). *Nested existential restrictions* $\exists r_1.\exists r_2.\dots.\exists r_n.C$ will sometimes also be written as $\exists r_1 r_2 \dots r_n.C$, where $r_1 r_2 \dots r_n$ is viewed as a word over the alphabet of role names, i.e., an element of N_R^* . As usual, concepts C are interpreted as sets $C^{\mathcal{I}}$ over some domain such that the semantics of the constructors is respected.

A *general concept inclusion (GCI)* is of the form $C \sqsubseteq D$ for concept descriptions C, D , and a general TBox is a finite set of GCIs. An interpretation \mathcal{I} *satisfies* such a GCI if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it is a *model* of the general TBox \mathcal{T} if it satisfies all GCIs in \mathcal{T} . Subsumption asks whether a given GCI $C \sqsubseteq D$ follows from a general TBox \mathcal{T} , i.e. whether every model of \mathcal{T} satisfies $C \sqsubseteq D$. In this

case we say C is *subsumed* by D w.r.t. \mathcal{T} and write $C \sqsubseteq_{\mathcal{T}} D$. Subsumption in \mathcal{EL} w.r.t. a general TBox is known to be decidable in polynomial time [9].

An \mathcal{EL} -concept description is an *atom* if it is an existential restriction or a concept name. The atoms of an \mathcal{EL} -concept description C are the subdescriptions of C that are atoms, and the top-level atoms of C are the atoms occurring in the top-level conjunction of C . Obviously, any \mathcal{EL} -concept description is the conjunction of its top-level atoms, where the empty conjunction corresponds to \top . The atoms of a general TBox \mathcal{T} are the atoms of all the concept descriptions occurring in \mathcal{T} .

We say that a subsumption between two atoms is *structural* if their top-level structure is compatible. To be more precise, we define structural subsumption between atoms as follows: the atom C is *structurally subsumed* by the atom D w.r.t. \mathcal{T} ($C \sqsubseteq_{\mathcal{T}}^s D$) iff either (i) $C = D$ is a concept name, or (ii) $C = \exists r.C'$, $D = \exists r.D'$, and $C' \sqsubseteq_{\mathcal{T}} D'$. It is easy to see that subsumption w.r.t. \emptyset between two atoms implies structural subsumption w.r.t. \mathcal{T} , which in turn implies subsumption w.r.t. \mathcal{T} . The unification algorithm in [2] and the one presented below crucially depend on the following characterization of subsumption:

Lemma 1. *Let \mathcal{T} be a general TBox and $C_1, \dots, C_n, D_1, \dots, D_m$ atoms. Then $C_1 \sqcap \dots \sqcap C_n \sqsubseteq_{\mathcal{T}} D_1 \sqcap \dots \sqcap D_m$ iff for every $j \in \{1, \dots, m\}$*

1. *there is an index $i \in \{1, \dots, n\}$ such that $C_i \sqsubseteq_{\mathcal{T}}^s D_j$, or*
2. *there are atoms A_1, \dots, A_k, B of \mathcal{T} ($k \geq 0$) such that*
 - a) $A_1 \sqcap \dots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$,
 - b) *for every $\eta \in \{1, \dots, k\}$ there is $i \in \{1, \dots, n\}$ with $C_i \sqsubseteq_{\mathcal{T}}^s A_\eta$, and*
 - c) $B \sqsubseteq_{\mathcal{T}}^s D_j$.

Our proof of this lemma in [1] is based on a new Gentzen-style proof calculus for subsumption w.r.t. a general TBox, which is similar to the one developed in [10] for subsumption w.r.t. cyclic and general TBoxes.

As mentioned in the introduction, our unification algorithm is complete only for general TBoxes that satisfy a certain restriction on cycles.

Definition 2. *The general TBox \mathcal{T} is called cycle-restricted iff there is no nonempty word $w \in N_R^+$ and \mathcal{EL} -concept description C such that $C \sqsubseteq_{\mathcal{T}} \exists w.C$.*

In [1] we show that a given general TBox can easily be tested for cycle-restrictedness. The main idea is that it is sufficient to consider the cases where C is a concept name or \top .

Lemma 3. *Let \mathcal{T} be a general TBox. It can be decided in time polynomial in the size of \mathcal{T} whether \mathcal{T} is cycle-restricted or not.*

3 Unification in \mathcal{EL} w.r.t. General TBoxes

We partition the set N_C into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). A *substitution* σ maps every concept variable to an \mathcal{EL} -concept description. It is extended to concept descriptions in the usual way:

- $\sigma(A) := A$ for all $A \in N_c \cup \{\top\}$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$ and $\sigma(\exists r.C) := \exists r.\sigma(C)$.

An \mathcal{EL} -concept description C is *ground* if it does not contain variables. Obviously, a ground concept description is not modified by applying a substitution. A general TBox is *ground* if it does not contain variables.

Definition 4. Let \mathcal{T} be a general TBox that is ground. An \mathcal{EL} -unification problem w.r.t. \mathcal{T} is a finite set $\Gamma = \{C_1 \sqsubseteq^? D_1, \dots, C_n \sqsubseteq^? D_n\}$ of subsumptions between \mathcal{EL} -concept descriptions. A substitution σ is a unifier of Γ w.r.t. \mathcal{T} if σ solves all the subsumptions in Γ , i.e. if $\sigma(C_1) \sqsubseteq_{\mathcal{T}} \sigma(D_1), \dots, \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(D_n)$. We say that Γ is unifiable w.r.t. \mathcal{T} if it has a unifier.

Note that we have restricted the background general TBox \mathcal{T} to be ground. This is not without loss of generality. If \mathcal{T} contained variables, then we would need to apply the substitution also to its GCIs, and instead of requiring $\sigma(C_i) \sqsubseteq_{\mathcal{T}} \sigma(D_i)$ we would thus need to require $\sigma(C_i) \sqsubseteq_{\sigma(\mathcal{T})} \sigma(D_i)$, which would change the nature of the problem considerably (see [1] for a more detailed discussion).

Preprocessing To simplify the description of the algorithm, it is convenient to first normalize the TBox and the unification problem appropriately. An atom is called *flat* if it is a concept name or an existential restriction of the form $\exists r.A$ for a concept name A . The general TBox \mathcal{T} is called *flat* if it contains only GCIs of the form $A \sqcap B \sqsubseteq C$, where A, B are flat atoms or \top and C is a flat atom. The unification problem Γ is called *flat* if it contains only flat subsumptions of the form $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? D$, where $n \geq 0$ and C_1, \dots, C_n, D are flat atoms.¹

Let Γ be a unification problem and \mathcal{T} a general TBox. By introducing auxiliary variables and concept names, respectively, Γ and \mathcal{T} can be transformed in polynomial time into a flat unification problem Γ' and a flat general TBox \mathcal{T}' such that the unifiability status remains unchanged, i.e., Γ has a unifier w.r.t. \mathcal{T} iff Γ' has a unifier w.r.t. \mathcal{T}' . In addition, if \mathcal{T} was cycle-restricted, then so is \mathcal{T}' (see [1] for details). Thus, we can assume without loss of generality that the input unification problem and general TBox are flat.

Local Unifiers The main idea underlying the “in NP” results in [4,2] is to show that any \mathcal{EL} -unification problem that is unifiable has a so-called local unifier. Let \mathcal{T} be a flat cycle-restricted TBox and Γ a flat unification problem. The atoms of Γ are the atoms of all the concept descriptions occurring in Γ . We define

$$\begin{aligned} \text{At} &:= \{C \mid C \text{ is an atom of } \mathcal{T} \text{ or of } \Gamma\} \text{ and} \\ \text{At}_{\text{nv}} &:= \text{At} \setminus N_v \quad (\text{non-variable atoms}). \end{aligned}$$

Every assignment S of subsets S_X of At_{nv} to the variables X in N_v induces the following relation $>_S$ on N_v : $>_S$ is the transitive closure of

$$\{(X, Y) \in N_v \times N_v \mid Y \text{ occurs in an element of } S_X\}.$$

¹ If $n = 0$, then we have an empty conjunction on the left-hand side, which as usual stands for \top .

We call the assignment S *acyclic* if $>_S$ is irreflexive (and thus a strict partial order). Any acyclic assignment S induces a unique substitution σ_S , which can be defined by induction along $>_S$:

- If $X \in N_v$ is minimal w.r.t. $>_S$, then we define $\sigma_S(X) := \prod_{D \in S_X} D$.
- Assume that $\sigma(Y)$ is already defined for all Y such that $X >_S Y$. Then we define $\sigma_S(X) := \prod_{D \in S_X} \sigma_S(D)$.

We call a substitution σ *local* if it is of this form, i.e., if there is an acyclic assignment S such that $\sigma = \sigma_S$. If the unifier σ of Γ w.r.t. \mathcal{T} is a local substitution, then we call it a *local unifier* of Γ w.r.t. \mathcal{T} .

The main technical result shown in [2] is that any unifiable \mathcal{EL} -unification problem w.r.t. a cycle-restricted TBox has a local unifier. This yields the following brute-force unification algorithm for \mathcal{EL} w.r.t. cycle-restricted TBoxes: first guess an acyclic assignment S , and then check whether the induced local substitution σ_S solves Γ . As shown in [2], this algorithm runs in nondeterministic polynomial time. NP-hardness follows from the fact that already unification in \mathcal{EL} w.r.t. the empty TBox is NP-hard [4].

4 A Goal-Oriented Unification Algorithm

The brute-force algorithm is not practical since it blindly guesses an acyclic assignment and only afterwards checks whether the guessed assignment induces a unifier. We now introduce a more goal-oriented unification algorithm, in which nondeterministic decisions are only made if they are triggered by “unsolved parts” of the unification problem. In addition, failure due to wrong guesses can be detected early. Any non-failing run of the algorithm produces a unifier, i.e., there is no need for checking whether the assignment computed by this run really produces a unifier. This goal-oriented algorithm generalizes the algorithm for unification in \mathcal{EL} w.r.t. the empty TBox introduced in [6], though the rules look quite different because in the present paper we consider unification problems that consist of subsumptions whereas in [6] we considered equivalences.

We assume without loss of generality that the cycle-restricted TBox \mathcal{T} and the unification problem Γ_0 are flat. Given \mathcal{T} and Γ_0 , the sets At and At_{nv} are defined as above. Starting with Γ_0 , the algorithm maintains a current unification problem Γ and a current acyclic assignment S , which initially assigns the empty set to all variables. In addition, for each subsumption in Γ it maintains the information on whether it is *solved* or not. Initially, all subsumptions are unsolved, except those with a variable on the right-hand side. Rules are applied only to unsolved subsumptions. A (non-failing) rule application does the following:

- it solves exactly one unsolved subsumption,
- it may extend the current assignment S , and
- it may introduce new flat subsumptions built from elements of At .

Each rule application that extends S_X additionally *expands* Γ w.r.t. X as follows: every subsumption $\mathfrak{s} \in \Gamma$ of the form $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? X$ is *expanded* by adding the subsumption $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? A$ to Γ for every $A \in S_X$.

Eager Ground Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? D$ if it is ground.
Action: If $C_1 \sqcap \dots \sqcap C_n \sqsubseteq_{\mathcal{T}} D$ does not hold, the rule application fails. Otherwise, \mathfrak{s} is marked as *solved*.

Eager Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? D$ if either
– there is $i \in \{1, \dots, n\}$ such that $C_i = D$ or $C_i = X \in N_v$ and $D \in S_X$, or
– D is ground and $\sqcap \mathcal{G} \sqsubseteq_{\mathcal{T}} D$ holds, where \mathcal{G} is the set of all ground atoms in $\{C_1, \dots, C_n\} \cup \bigcup_{X \in \{C_1, \dots, C_n\} \cap N_v} S_X$.
Action: Its application marks \mathfrak{s} as *solved*.

Eager Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? D$ if there is $i \in \{1, \dots, n\}$ with $C_i = X \in N_v$ and $\{C_1, \dots, C_n\} \setminus \{X\} \subseteq S_X$.
Action: Its application adds D to S_X . If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. X and \mathfrak{s} is marked as *solved*.

Fig. 1. The eager rules of the unification algorithm.

Subsumptions are only added if they are not already present in Γ . If a new subsumption is added to Γ , either by a rule application or by expansion of Γ , then it is initially designated unsolved, except if it has a variable on the right-hand side. Once a subsumption is in Γ , it will not be removed. Likewise, if a subsumption in Γ is marked as solved, then it will not become unsolved later.

If a subsumption is marked as solved, this does not mean that it is already solved by the substitution induced by the current assignment. It may be the case that the task of satisfying the subsumption was deferred to solving other subsumptions which are “smaller” than the given subsumption in a well-defined sense. The task of solving a subsumption whose right-hand side is a variable is deferred to solving the subsumptions introduced by expansion.

The rules of the algorithm consist of the three *eager* rules Eager Ground Solving, Eager Solving, and Eager Extension (see Figure 1), and several *nondeterministic* rules (see Figures 2 and 3). Eager rules are applied with higher priority than nondeterministic rules. Among the eager rules, Eager Ground Solving has the highest priority, then comes Eager Solving, and then Eager Extension.

Algorithm 5. Let Γ_0 be a flat \mathcal{EL} -unification problem. We set $\Gamma := \Gamma_0$ and $S_X := \emptyset$ for all $X \in N_v$. While Γ contains an unsolved subsumption, apply the steps (1) and (2). Once all subsumptions are solved, return the substitution σ induced by the current assignment.

- (1) **Eager rule application:** If some eager rules apply to an unsolved subsumption \mathfrak{s} in Γ , apply one of highest priority. If the rule application fails, then return “not unifiable”.
- (2) **Nondeterministic rule application:** If no eager rule is applicable, let \mathfrak{s} be an unsolved subsumption in Γ . If one of the nondeterministic rules applies to \mathfrak{s} , nondeterministically choose one of these rules and apply it. If none of these rules apply to \mathfrak{s} or the rule application fails, then return “not unifiable”.

Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? \exists s.D'$ if there is at least one index $i \in \{1, \dots, n\}$ with $C_i = \exists s.C'$.

Action: Its application chooses such an index i , adds the subsumption $C' \sqsubseteq^? D'$ to Γ , expands it w.r.t. D' if D' is a variable, and marks \mathfrak{s} as *solved*.

Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? D$ if there is at least one $i \in \{1, \dots, n\}$ with $C_i \in N_v$.

Action: Its application chooses such an i and adds D to S_{C_i} . If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. C_i and \mathfrak{s} is marked as *solved*.

Fig. 2. The nondeterministic rules *Decomposition* and *Extension*.

In step (2), the choice which unsolved subsumption to consider next is don't care nondeterministic. However, choosing which rule to apply to the chosen subsumption is don't know nondeterministic. Additionally, the application of nondeterministic rules requires don't know nondeterministic guessing.

The *eager rules* are mainly there for optimization purposes, i.e., to avoid nondeterministic choices if a deterministic decision can be made. For example, a ground subsumption, as considered in the *Eager Ground Solving* rule, either follows from the TBox, in which case any substitution solves it, or it does not, in which case it does not have a solution. This condition can be checked in polynomial time using the polynomial time subsumption algorithm for \mathcal{EL} [9]. In the case considered in the *Eager Solving* rule, the substitution induced by the current assignment already solves the subsumption. In fact, if the first (second) condition of the rule is satisfied, then the first (second) condition of Lemma 1 applies. The *Eager Extension* rule solves a subsumption that contains only a variable X and some elements of S_X on the left-hand side. The rule is motivated by the following observation: for any assignment S' extending the current assignment, the induced substitution σ' satisfies $\sigma'(X) \equiv \sigma'(C_1) \sqcap \dots \sqcap \sigma'(C_n)$. Thus, if S'_X contains D , then $\sigma'(X) \sqsubseteq_{\mathcal{T}} \sigma'(D)$, and σ' solves the subsumption. Conversely, if σ' solves the subsumption, then $\sigma'(X) \sqsubseteq_{\mathcal{T}} \sigma'(D)$, and thus adding D to S'_X yields an equivalent induced substitution.

The *nondeterministic rules* only come into play if no eager rules can be applied. In order to solve an unsolved subsumption $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? D$, we consider the two conditions of Lemma 1. Regarding the first condition, which is addressed by the rules *Decomposition* and *Extension*, assume that γ is induced by an acyclic assignment S . To satisfy the first condition of the lemma with γ , the atom $\gamma(D)$ must subsume a top-level atom in $\gamma(C_1) \sqcap \dots \sqcap \gamma(C_n)$. This atom can either be of the form $\gamma(C_i)$ for an atom C_i , or it can be of the form $\gamma(C)$ for an atom $C \in S_{C_i}$ and a variable C_i . In the second case, the atom C can either already be in S_{C_i} or it can be put into S_{C_i} by an application of the *Extension* rule. The *Mutation rules* cover the second condition in Lemma 1. For example, let us analyze in detail how *Mutation 1* ensures that all the requirements of the second condition of Lemma 1 are satisfied. Whenever this condition requires a structural

Mutation 1:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? D$ if $n > 1$ and there are atoms A_1, \dots, A_k, B of \mathcal{T} such that $A_1 \sqcap \dots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$ holds.

Action: Its application chooses such atoms, marks \mathfrak{s} as *solved*, and generates the following subsumptions:

- it chooses for each $\eta \in \{1, \dots, k\}$ an $i \in \{1, \dots, n\}$ and adds the new subsumption $C_i \sqsubseteq^? A_\eta$ to Γ ,
- it adds $B \sqsubseteq^? D$ to Γ .

Mutation 2:

Condition: This rule applies to $\mathfrak{s} = \exists r.X \sqsubseteq^? D$ if X is a variable, D is ground, and there are atoms $\exists r.A_1, \dots, \exists r.A_k$ of \mathcal{T} such that $\exists r.A_1 \sqcap \dots \sqcap \exists r.A_k \sqsubseteq_{\mathcal{T}} D$ holds.

Action: Its application chooses such atoms, adds A_1, \dots, A_k to S_X , expands Γ w.r.t. X , and marks \mathfrak{s} as *solved*.

Mutation 3:

Condition: This rule applies to $\mathfrak{s} = \exists r.X \sqsubseteq^? \exists s.Y$ if X and Y are variables, and there are atoms $\exists r.A_1, \dots, \exists r.A_k, \exists s.B$ of \mathcal{T} with $\exists r.A_1 \sqcap \dots \sqcap \exists r.A_k \sqsubseteq_{\mathcal{T}} \exists s.B$.

Action: Its application chooses such atoms, marks \mathfrak{s} as *solved*, and generates the following subsumptions:

- it adds A_1, \dots, A_k to S_X and expands Γ w.r.t. X ,
- it adds the subsumption $B \sqsubseteq^? Y$ to Γ and expands it w.r.t. Y .

Mutation 4:

Condition: This rule applies to $\mathfrak{s} = C \sqsubseteq^? \exists s.Y$ if C is a ground atom or \top , Y is a variable, and there is an atom $\exists s.B$ of \mathcal{T} such that $C \sqsubseteq_{\mathcal{T}} \exists s.B$ holds.

Action: Its application chooses such an atom, adds the new subsumption $B \sqsubseteq^? Y$ to Γ , expands this subsumption w.r.t. Y , and marks \mathfrak{s} as *solved*.

Fig. 3. The nondeterministic *Mutation* rules of the unification algorithm.

subsumption $\gamma(E) \sqsubseteq_{\mathcal{T}}^{\mathfrak{s}} \gamma(F)$ to hold for a (hypothetical) unifier γ of Γ , the rule creates the new subsumption $E \sqsubseteq^? F$, which has to be solved later on. This way, the rule ensures that the substitution built by the algorithm actually satisfies the conditions of the lemma. To check the subsumption $A_1 \sqcap \dots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$, the rule again employs a polynomial-time subsumption algorithm.

The *other mutation rules* follow the same idea, but they implicitly apply one or more Decomposition or Eager Extension rules after mutation. This ensures that the generated subsumptions are “smaller” than the subsumption that triggers their introduction.

Soundness We will show that, if Algorithm 5 returns a substitution σ on input Γ_0 , then σ is a unifier of Γ_0 w.r.t. \mathcal{T} . In the following, let S be the final acyclic assignment computed by a non-failing run of Algorithm 5 on input Γ_0 , and σ the substitution induced by S . By $\widehat{\Gamma}$ we denote the final set of subsumptions computed by this run, i.e., the original subsumptions of Γ_0 together with the

new ones generated by rule applications. To show that σ solves all subsumptions in $\widehat{\Gamma}$, we use well-founded induction [8] on the well-founded order \succ on $\widehat{\Gamma}$:

Definition 6. Let $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? C_{n+1} \in \widehat{\Gamma}$.

- \mathfrak{s} is small if $n = 1$ and C_1 is ground or C_{n+1} is ground.
- We define $m(\mathfrak{s}) := (m_1(\mathfrak{s}), m_2(\mathfrak{s}), m_3(\mathfrak{s}))$, where
 - $m_1(\mathfrak{s}) := 0$ if \mathfrak{s} is small, and $m_1(\mathfrak{s}) := 1$ otherwise;
 - $m_2(\mathfrak{s}) := X$ if $C_{n+1} = X$ or $C_{n+1} = \exists r.X$ for a variable X and some $r \in N_R$, and $m_2(\mathfrak{s}) := \perp$ otherwise;
 - $m_3(\mathfrak{s}) := \max\{rd(\sigma(C_i)) \mid i \in \{1, \dots, n+1\}\}$ where rd yields the role depth of a concept description, i.e., the maximal nesting of existential restrictions.
- The strict partial order \succ on such triples is the lexicographic order, where the first and the third component are compared w.r.t. the normal order $>$ on natural numbers. The variables in the second component are compared w.r.t. the relation $>_S$ induced by S , and \perp is smaller than any variable.
- We extend \succ to $\widehat{\Gamma}$ by setting $\mathfrak{s}_1 \succ \mathfrak{s}_2$ iff $m(\mathfrak{s}_1) \succ m(\mathfrak{s}_2)$.

As the lexicographic product of well-founded strict partial orders is again well-founded [8], \succ is a well-founded strict partial order on $\widehat{\Gamma}$.

Lemma 7. σ is an \mathcal{EL} -unifier of $\widehat{\Gamma}$ w.r.t. \mathcal{T} , and thus also of its subset Γ_0 .

Proof. Let $\mathfrak{s} \in \widehat{\Gamma}$ and assume that σ solves all subsumptions $\mathfrak{s}' \in \widehat{\Gamma}$ with $\mathfrak{s}' \prec \mathfrak{s}$.

- If \mathfrak{s} has a *non-variable atom as its right-hand side*, then it was initially marked as unsolved and must have been marked solved by a successful rule application. As an example, we consider the application of the Decomposition rule (the other rules can be treated similarly [1]). Then \mathfrak{s} is of the form $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? \exists s.D'$ with $C_i = \exists s.C'$ for some $i \in \{1, \dots, n\}$ and we have $\mathfrak{s}' = C' \sqsubseteq^? D' \in \widehat{\Gamma}$. We will show that $\mathfrak{s} \succ \mathfrak{s}'$ holds. By induction, this implies that σ solves \mathfrak{s}' , and by Lemma 1 thus also \mathfrak{s} .
Observe first that $m_2(\mathfrak{s}) = m_2(\mathfrak{s}')$ since either $\exists s.D'$ and D' contain the same variable or are both ground. We now make a case distinction based on $m_1(\mathfrak{s}')$. If \mathfrak{s}' is small, then \mathfrak{s} is either non-small, i.e. $m_1(\mathfrak{s}) > m_1(\mathfrak{s}')$, or small and of the form $\exists s.C' \sqsubseteq^? \exists s.D'$. In the second case, we have $m_1(\mathfrak{s}) = m_1(\mathfrak{s}')$ and $m_3(\mathfrak{s}) > m_3(\mathfrak{s}')$. If \mathfrak{s}' is non-small, then both C' and D' are variables, and thus \mathfrak{s} is also non-small, which yields $m_1(\mathfrak{s}) = m_1(\mathfrak{s}')$. Furthermore, the maximal role depth obviously decreases when going from \mathfrak{s} to \mathfrak{s}' , and thus $m_3(\mathfrak{s}) > m_3(\mathfrak{s}')$. In all cases we have shown $m(\mathfrak{s}) \succ m(\mathfrak{s}')$, i.e., $\mathfrak{s} \succ \mathfrak{s}'$.
- If \mathfrak{s} has a *variable as its right-hand side*, it is of the form $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? X$ and for every $A \in S_X$ there is a subsumption $\mathfrak{s}_A = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? A$ in $\widehat{\Gamma}$. If \mathfrak{s} is small, then $n = 1$ and C_1 is ground, and thus the subsumptions \mathfrak{s}_A are also small. Thus, we have $m_1(\mathfrak{s}) \geq m_1(\mathfrak{s}_A)$ for every $A \in S_X$. Furthermore, we have $m_2(\mathfrak{s}) > m_2(\mathfrak{s}_A)$ since A is ground or contains a variable on which X depends. This yields $\mathfrak{s} \succ \mathfrak{s}_A$, and thus by induction $\sigma(C_1) \sqcap \dots \sqcap \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(A)$ for every $A \in S_X$, which implies that $\sigma(C_1) \sqcap \dots \sqcap \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(X)$ by the definition of σ . \square

Completeness Assume that Γ_0 is unifiable w.r.t. \mathcal{T} and let γ be a ground unifier of Γ_0 w.r.t. \mathcal{T} . We use this unifier to guide the application of the nondeterministic rules such that Algorithm 5 does not fail. The following invariants for Γ and S will be maintained:

- (I) γ is a unifier of Γ .
- (II) For all $B \in S_X$ we have $\gamma(X) \sqsubseteq_{\mathcal{T}} \gamma(B)$.

Since S_X is initialized to \emptyset for all variables $X \in N_v$ and Γ is initialized to Γ_0 , these invariants are satisfied after the initialization of the algorithm.

The invariants immediately rule out one cause of failure for the algorithm, namely that the current assignment becomes cyclic. This is the only place in the whole proof where our assumption on cycle-restrictedness of \mathcal{T} is needed.

Lemma 8. *If invariant (II) is satisfied, then the current assignment S is acyclic.*

The proofs of this and of the next lemma can be found in [1].

Lemma 9. *Assume that the current set of subsumptions Γ and the current assignment S satisfy the invariants (I) and (II), and let $\mathfrak{s} \in \Gamma$ be unsolved.*

1. *If an eager rule applies to \mathfrak{s} , then its application does not fail and the resulting set Γ' and assignment S' also satisfy the invariants (I) and (II).*
2. *If no eager rule applies to \mathfrak{s} , then there is a nondeterministic rule that can successfully be applied to \mathfrak{s} such that the resulting set Γ' and assignment S' also satisfy the invariants (I) and (II).*

An immediate consequence of this lemma is that, if Γ_0 is unifiable, then there is a non-failing run of Algorithm 5 on Γ_0 during which the invariants (I) and (II) are satisfied. Together with the fact that every run of the algorithm terminates (see below), this shows completeness, i.e., whenever Γ_0 has a unifier w.r.t. \mathcal{T} , the algorithm computes one.

Termination Consider a run of Algorithm 5. It is easy to show that any subsumption encountered during this run falls into one of the following categories:

1. subsumptions from Γ_0 ;
2. subsumptions created by expansion from Γ_0 : these are of the form $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? A$ for a subsumption $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? X \in \Gamma_0$ and $A \in \text{At}_{\text{nv}}$;
3. subsumptions of the form $C \sqsubseteq^? D$ for $C, D \in \text{At}$.

Since the cardinality of At is polynomially bounded by the size of Γ_0 and \mathcal{T} , there are only polynomially many subsumptions of this form. Rules are only applicable to subsumptions that are marked unsolved, and the application of a rule marks at least one subsumption as solved. Thus, only polynomially many rules can be applied during the run. In addition, each rule application takes only polynomial time. This shows that every run of the algorithm terminates in polynomial time.

Theorem 10. *Algorithm 5 is an NP-decision procedure for unifiability in \mathcal{EL} w.r.t. cycle-restricted TBoxes.*

5 Conclusions

We have presented a goal-oriented NP-algorithm for unification in \mathcal{EL} w.r.t. cycle-restricted TBoxes. In [3], we developed a reduction of this problem to a propositional satisfiability problem (SAT), which is based on a characterization of subsumption different from the one in Lemma 1. Though clearly better than the brute-force algorithm introduced in [2], both algorithms suffer from a high degree of nondeterminism introduced by having to guess atoms and GCIs from the underlying cycle-restricted TBox. We have to find optimizations to tackle this problem before an implementation becomes feasible.

On the theoretical side, the main topic for future research is to consider unification w.r.t. unrestricted general TBoxes. In order to generalize the brute-force algorithm in this direction, we need to find a more general notion of locality. Starting with the goal-oriented algorithm, the idea would be not to fail when a cyclic assignment is generated, but rather to add rules that can break such cycles, similar to what is done in procedures for general E -unification [11].

References

1. Baader, F., Borgwardt, S., Morawska, B.: Unification in the description logic \mathcal{EL} w.r.t. cycle-restricted TBoxes. LTCS-Report 11-05, Chair of Automata Theory, TU Dresden, Germany (2011), see <http://lat.inf.tu-dresden.de/research/reports.html>.
2. Baader, F., Borgwardt, S., Morawska, B.: Extending unification in \mathcal{EL} towards general TBoxes. In: Proc. KR'12. AAAI Press (2012), short paper. To appear.
3. Baader, F., Borgwardt, S., Morawska, B.: SAT encoding of unification in \mathcal{ELH}_{R+} w.r.t. cycle-restricted ontologies. LTCS-Report 12-02, Chair for Automata Theory, TU Dresden (2012), see <http://lat.inf.tu-dresden.de/research/reports.html>. A short version of this report has been submitted to a conference.
4. Baader, F., Morawska, B.: Unification in the description logic \mathcal{EL} . In: Proc. RTA'09. LNCS, vol. 5595, pp. 350–364. Springer (2009)
5. Baader, F., Morawska, B.: SAT encoding of unification in \mathcal{EL} . In: Proc. LPAR'10. LNCS, vol. 6397, pp. 97–111. Springer (2010)
6. Baader, F., Morawska, B.: Unification in the description logic \mathcal{EL} . Log. Meth. Comput. Sci. 6(3) (2010)
7. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. Symb. Comput. 31(3), 277–305 (2001)
8. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, United Kingdom (1998)
9. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: Proc. ECAI'04. pp. 298–302. IOS Press (2004)
10. Hofmann, M.: Proof-theoretic approach to description-logic. In: Proc. LICS'05. pp. 229–237. IEEE Press (2005)
11. Morawska, B.: General E -unification with eager variable elimination and a nice cycle rule. J. Autom. Reasoning 39(1), 77–106 (2007)

Diversity of Reason: Equivalence Relations over Description Logic Explanations

Samantha Bail, Bijan Parsia, Ulrike Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL
{bails,bparsia,sattler@cs.man.ac.uk}

Abstract. Given the high expressivity of modern ontology languages, such as OWL, there is the possibility for great diversity in the logical content of ontologies. Informally, this can be seen by the constant evolution of reasoners to deal with new sorts of content and the range of optimisations reasoners need in order to be competitive. More formally, the fact that many naturally occurring entailments have multiple justifications (i.e., minimal entailing subsets) indicates that ontologies often overdetermine their consequences, indicating a diversity in supporting reasons. However, the multiplicity of justifications might be due mostly to diverse *material*, not *formal*, grounds for an entailment. That is, the logical form of these multiple reasons could be less diverse than their numbers suggest.

In the present paper, we introduce and explore several equivalence relations over justifications for entailments of OWL ontologies. These equivalence relations range from strict isomorphism to a looser notions which cover similarities between justifications containing different concept expressions or possibly different numbers of axioms. We survey a corpus of ontologies from the bio-medical domain and find that large numbers of justifications can often be reduced to a significantly smaller set of justifications which are isomorphic with respect to one of the given definitions.

1 Introduction

Since its standardisation by the W3C in 2004, the Web Ontology Language OWL has been used to represent domain knowledge from diverse areas, such as medicine and general biology. OWL 2¹ is based on the highly expressive Description Logic *SRIQ* [10] as underlying formalism: An OWL 2 ontology corresponds to a set of *SRIQ* axioms.

Justifications, minimal entailing subsets of an OWL ontology, provide helpful and easy-to-understand explanation support when repairing unwanted entailments in the ontology debugging process. While previous research has focused on the issue of making individual justifications easier to understand, only little attention has been paid to the occurrence of *multiple justifications* other than

¹ <http://www.w3.org/TR/owl2-syntax>

the computational problems they often imply. An entailment of an OWL ontology can have a large number of justifications (potentially exponential in the number of axioms in the ontology), with up to several thousands found in large real-life ontologies.

When encountering justifications for a finite set of entailments of an ontology (e.g. the set of entailed atomic subsumptions), we are often faced with a seemingly large and diverse body of reasons why these entailments hold. On closer inspection, however, we frequently find that multiple justifications for different entailments are identical sets of axioms, or that justifications are structurally identical and only diverge in the concept, role, and individual names they use.

In order to draw a clearer picture of the logical diversity of a corpus of justifications, we need to take into account these similarities between justifications. Grouping the set of justifications into subsets according to some similarity measure helps structuring the unsorted pool of justifications. Rather than trying to understand each individual justification, users can focus on understanding the *template* of a particular subset of justifications. This can lead to significantly fewer justifications that the user has to deal with, and therefore to a reduction in user effort.

A well-known syntactical equivalence relation in OWL is *structural equivalence*. The OWL Structural Specification² states the condition for two OWL objects (named concepts, roles, or instances, complex expressions, or OWL axioms) to be structurally equivalent. In short, it defines the objects to be equivalent if they contain the same names and constructors, regardless of ordering and repetition (in an unordered association). The OWL API,³ a Java API which is used to manipulate OWL ontologies, implements this notion of structural equivalence by default.

Justification isomorphism [5] was first introduced in a study of the cognitive complexity of justifications: Two justifications are isomorphic if they are structurally identical,⁴ i.e. the axioms contain the same concept expressions and only differ in the concept, role and individual names.

While justification isomorphism helps to eliminate the effects of diverging concept, role, and individual names, we can also identify types of justifications which may be considered to be very similar despite their use of different constructors:

Example 1

$$\begin{aligned} \mathcal{J}_1 &= \{A \sqsubseteq B \sqcap C, B \sqcap C \sqsubseteq D\} \models A \sqsubseteq D \\ \mathcal{J}_2 &= \{A \sqsubseteq \exists r.C, \exists r.C \sqsubseteq D\} \models A \sqsubseteq D \end{aligned}$$

In this example, the semantics of the complex concept expressions $B \sqcap C$ in \mathcal{J}_1 and $\exists r.C$ in \mathcal{J}_2 are not relevant for the respective entailment; their occurrences in the

² <http://www.w3.org/TR/owl2-syntax>

³ <http://owlapi.sourceforge.net>

⁴ Modulo structural equivalence, which takes into account redundant expressions and the commutativity of constructors.

justifications and their entailments can be replaced by freshly generated atomic concept names without affecting the entailment relation. Such a substitution in turn would make the two justifications isomorphic.

Likewise, justifications of different lengths may be considered similar if their general structure of reasoning is identical:

Example 2

$$\begin{aligned} \mathcal{J}_1 &= \{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C \\ \mathcal{J}_2 &= \{A \sqsubseteq B, B \sqsubseteq C, C \sqsubseteq D\} \models A \sqsubseteq D \end{aligned}$$

These two justifications clearly require the same form of reasoning from a user. Strict isomorphism only applies to justifications which contain the same number of axioms; it does not cover situations like the above. However, for the purpose of structuring sets of justifications and analysing the logical diversity of a corpus of justifications, capturing those kinds of similarities illustrated in the above examples would be highly desirable.

These examples motivate a looser notion of isomorphism, which allows us to identify justifications as equivalent if they require the same reasoning mechanisms, regardless of size, signature and logical constructors used. In the present paper, we introduce two new types of equivalence relations based on matching subexpressions and lemmas, and show how these extended relations are applied to a corpus of justifications from the bio-medical domain.

2 Preliminaries

Justifications in OWL We assume the reader to be familiar with OWL and the underlying Description Logic *SR_QI_Q*. In what follows, A, B, \dots denote concept names in an ontology \mathcal{O} , r, s role names, and $sig(\alpha)$ denotes the set of concept, role, and individual names in an OWL axiom α .

Justifications [13,11] are a popular type of explanation for entailments of OWL ontologies. A justification is defined as a minimal subset of an ontology \mathcal{O} that causes an entailment η to hold:

Definition 1 (Justification) \mathcal{J} is a justification for $\mathcal{O} \models \eta$ if $\mathcal{J} \subseteq \mathcal{O}, \mathcal{J} \models \eta$ and, for all $\mathcal{J}' \subset \mathcal{J}$, it holds that $\mathcal{J}' \not\models \eta$.

For every axiom which is asserted in the ontology, the axiom itself naturally is a justification. We call an entailment which has only itself as a justification a *self-supporting* entailment, and the justification a *self-justification*.

It is important to note that a justification is always defined with respect to an entailment η . In the remainder of this paper we will therefore use the term *justification* to describe a justification-entailment pair (\mathcal{J}, η) where \mathcal{J} is a justification for η .

Justification Isomorphism Isomorphism between justifications was first introduced as a method to reduce the number of similar justifications when sampling from a large corpus to justifications of distinct types [5].

Definition 2 (Justification Isomorphism) *Two justifications (\mathcal{J}_1, η_1) , (\mathcal{J}_2, η_2) are isomorphic $((\mathcal{J}_1, \eta_1) \approx_i (\mathcal{J}_2, \eta_2))$ if there exists an injective renaming ϕ which maps concept, role, and individual names in \mathcal{J}_1 and η_1 to concept, role, and individual names in \mathcal{J}_2 and η_2 , respectively, such that $\phi(\mathcal{J}_1) = \mathcal{J}_2$ and $\phi(\eta_1) = \eta_2$.*

Example 3 (Isomorphic Justifications)

$$\begin{aligned} \mathcal{J}_1 &= \{A \sqsubseteq B \sqcap \exists r.C, B \sqcap \exists r.C \sqsubseteq D\} && \models A \sqsubseteq D \\ \mathcal{J}_2 &= \{E \sqsubseteq B \sqcap \exists s.F, B \sqcap \exists s.F \sqsubseteq D\} && \models E \sqsubseteq D \\ \phi &= \{A \mapsto E, C \mapsto F, r \mapsto s\} \end{aligned}$$

The relation \approx_i is symmetric, reflexive and transitive, from which it follows that \approx_i is an equivalence relation and thus partitions a set of justifications. Justification isomorphism preserves the numbers and types of axioms and subexpressions in the justifications:

1. $\mathcal{J}_1 \approx_i \mathcal{J}_2 \rightarrow |\mathcal{J}_1| = |\mathcal{J}_2|$
2. $\mathcal{J}_1 \approx_i \mathcal{J}_2 \rightarrow |\text{sig}(\mathcal{J}_1)| = |\text{sig}(\mathcal{J}_2)|$
3. The sets of logical constructs used in \mathcal{J}_1 and \mathcal{J}_2 coincide.

In the remainder of this paper, we may refer to the isomorphism defined above as *strict* isomorphism in order to distinguish it from the other equivalence relations.

3 Subexpression-Isomorphism

From the above definition of isomorphism it follows that only justifications which have the same number and types of axioms and subexpressions can be isomorphic. It is easy to see, however, that justifications can have a similar structure despite their use of different concept expressions, as demonstrated in Example 1. This motivates a notion of isomorphism which allows not only the mapping of concept names, but also that of subexpressions.

The idea of finding similarities between concepts in Description Logics has been widely explored in the work on *unification* and *matching*, e.g. [2,1,3], for the purpose of detecting redundant concept descriptions in knowledge bases. The aim of unification is to find a suitable substitution σ which maps atomic concepts in a concept term C to (possibly non-atomic) concepts in a concept term D such that the two terms are made equivalent.

While the basic idea behind extended subexpression-isomorphism is based on unification and matching, the concepts are not directly applicable to the given problem of matching justifications. We therefore introduce a *unifying justification* \mathcal{J} , which functions as the *template* for the isomorphic justifications:

Definition 3 (Subexpression-Isomorphism) *Two justifications (\mathcal{J}_1, η_1) , (\mathcal{J}_2, η_2) are s-isomorphic ($(\mathcal{J}_1, \eta_1) \approx_s (\mathcal{J}_2, \eta_2)$) if there exists a justification (\mathcal{J}, η) and two injective substitutions ϕ_1, ϕ_2 , such that $\phi_1(\mathcal{J}) = \mathcal{J}_1$, $\phi_2(\mathcal{J}) = \mathcal{J}_2$, $\phi_1(\eta) = \eta_1$, and $\phi_2(\eta) = \eta_2$. The mappings ϕ_1 and ϕ_2 map concept, role, and individual names in (\mathcal{J}, η) to subexpressions of (\mathcal{J}_1, η_1) and (\mathcal{J}_2, η_2) , respectively.*

It can be shown that the relation \approx_s is reflexive, transitive and symmetric; it is therefore an equivalence relation and thus partitions a set of justifications.

The substitutions ϕ_1, ϕ_2 are injective, but not surjective, as the set of subexpressions in the justifications \mathcal{J}_1 and \mathcal{J}_2 can be of higher arity than the set of concept names in the unifying justification \mathcal{J} (unless the justifications themselves contain no complex expressions).

S-isomorphism can easily be shown to be a more general case of strict isomorphism between two justifications \mathcal{J}_1 and \mathcal{J}_2 : The unifying justification \mathcal{J} is set to \mathcal{J}_1 , ϕ_1 is the identity relation *id* which maps \mathcal{J}_1 to itself, and ϕ_2 corresponds to the mapping ϕ , which maps concept, role, and individual names in \mathcal{J}_2 to the respective names in \mathcal{J}_1 . It follows that $\mathcal{J}_1 \approx_i \mathcal{J}_2 \rightarrow \mathcal{J}_1 \approx_s \mathcal{J}_2$.

In order to be s-isomorphic, the justifications may differ in the number of subexpressions. They must, however, have the same number of axioms: $\mathcal{J}_1 \approx_s \mathcal{J}_2 \rightarrow |\mathcal{J}_1| = |\mathcal{J}_2|$

While we focus on entailed atomic subsumptions in the present paper, we point out that s-isomorphism between justifications is not restricted to a specific axiom type as entailment, as the substitutions ϕ_1, ϕ_2 preserve all entailment relations regardless of the axiom type and constructor usage.

4 Lemma-Isomorphism

While s-isomorphism covers a number of justifications that can be regarded as equivalent due to them requiring the same type of reasoning to reach the entailment, it only applies to justifications which have the same number of axioms. This does not take into account cases where the justifications differ only marginally in some subset, but where the general reasoning may be regarded as similar nonetheless. We therefore introduce the notion of *lemma-isomorphism*, which extends subexpression-isomorphism with the substitution of subsets of justifications through intermediate entailments, so-called *lemmas* [7]. The general motivation behind lemma-isomorphism is demonstrated by the following example:

Example 4

$$\begin{array}{ll} \mathcal{J}_1 = \{A \sqsubseteq B, B \sqsubseteq C\} & \models A \sqsubseteq C \\ \mathcal{J}_2 = \{A \sqsubseteq B, B \sqsubseteq C, C \sqsubseteq D\} & \models A \sqsubseteq D \end{array}$$

It is straightforward to see that both \mathcal{J}_1 and \mathcal{J}_2 require the same type of reasoning from a human user. As the justifications only differ in the length of

the atomic subsumption chains that lead to the entailment, we can certainly consider them to be *similar* with respect to *some* similarity measure. However, the two justifications are not considered isomorphic with respect to the definitions for strict isomorphism or subexpression-isomorphism. We therefore introduce a new type of isomorphism which takes into account the fact that subsets of justifications can be replaced with intermediate entailments which follow from them.

Lemmas of OWL justifications have previously found use in the extension of justifications to *justification-oriented proofs* [9]. The following definitions introduce simplified variants of the definitions [7] of justification lemmas and lemmatisations. Please note that for the purpose of illustrating the effect of lemma-isomorphism, we will simplify the lemmatisations to a more specific type of lemmas in the next section.

Definition 4 (Lemma) *Let \mathcal{J} be a justification for an entailment η . A lemma of (\mathcal{J}, η) is an axiom λ for which there exists a subset $S \subseteq \mathcal{J}$ such that $S \models \lambda$. A summarising lemma of (\mathcal{J}, η) is a lemma λ for which there exists an $S \subseteq \mathcal{J}$ such that $\mathcal{J} \setminus S \cup \{\lambda\} \models \eta$ for $S \models \lambda$.*

Definition 5 (Lemmatisation) *Let (\mathcal{J}, η) be a justification, let $S_1 \dots S_k$ be subsets of \mathcal{J} , and let $\lambda_1 \dots \lambda_k$ be axioms satisfying $S_i \models \lambda_i$ for $i \in \{1, \dots, k\}$. Then the set $\mathcal{J}^\Lambda := (\mathcal{J} \setminus \bigcup S_i) \cup \bigcup \lambda_i$ is called a lemmatisation of \mathcal{J} if $\mathcal{J}^\Lambda \models \eta$.*

If clear from the context, a lemmatisation \mathcal{J}^Λ may also be called a *lemmatised justification*. Given the definitions for lemmatisations, we can now define lemma-isomorphism as an extension to subexpression-isomorphism:

Definition 6 (Lemma-isomorphism) *Two justifications (\mathcal{J}_1, η_1) , (\mathcal{J}_2, η_2) are ℓ -isomorphic ($(\mathcal{J}_1, \eta) \approx_\ell (\mathcal{J}_2, \eta)$) if there exist lemmatisations $\mathcal{J}_1^{\Lambda_1}$, $\mathcal{J}_2^{\Lambda_2}$ which are s -isomorphic: $\mathcal{J}_1^{\Lambda_1} \approx_s \mathcal{J}_2^{\Lambda_2}$.*

Lemma-isomorphism using arbitrary lemmas as defined above carries some undesirable properties: First, unlike the previously defined relations, it describes a relation which is not transitive. Further, the lemmatised justifications might differ strongly from the original justifications; in the most extreme case, the lemmatisation of a justification can be the entailment itself. We therefore have to introduce some constraints on the admissible lemmatisations in order to guarantee the transitivity of the isomorphism relation, as well as preserve the nature of the original justifications. In what follows we will focus on lemmatisations through *obvious steps* in justifications.

4.1 Lemmatisations and Obvious Steps

The notion of *obvious proof steps* [4,12] describes how proof steps which are intuitively *obvious* can be replaced with their conclusion (thereby shortening the proof) without omitting important information. We loosely base the lemma

restriction on this obviousness and select an example of obvious proof steps which commonly occur in DL justifications, namely chains of atomic subsumptions.

In atomic subsumption chains of the type $A_0 \sqsubseteq A_1, A_1 \sqsubseteq A_2 \dots A_{n-1} \sqsubseteq A_n$ only the relation between the subconcept A_0 in the first axiom and the superconcept A_n in the last axiom are relevant for understanding the subsumption chain; i.e. the step from the subconcept to the final superconcept is *obvious*. We can say that it is only important to understand *that* there is a connection between the subconcept and the final superconcept, but we do not need to know *what* this connection is. Therefore, it seems reasonable to substitute the chain with its conclusion in the form of a single axiom $A_0 \sqsubseteq A_n$. Restricting lemmatisations to atomic subsumption chains leads to the definition for a transitivity-preserving type of l-isomorphism:

Definition 7 (Transitivity-preserving l-isomorphism) *Two justifications $(\mathcal{J}_1, \eta_1), (\mathcal{J}_2, \eta_2)$ are lt-isomorphic $((\mathcal{J}_1, \eta) \approx_{lt} (\mathcal{J}_2, \eta))$ if there exist summarising lemmatisations $\mathcal{J}_1^{A_1}, \mathcal{J}_2^{A_2}$ which are s-isomorphic, and every $S_i \subseteq \mathcal{J}_i$ where $S_i \models \lambda_i \in A_1 (\lambda_i \in A_2, \text{ respectively})$ is of type $\{A_i \sqsubseteq A_{i+1} \mid i \in \{0, \dots, n\}\}$ where n is the number of axioms in the respective chain of subsumption axioms, and A_i a concept name.*

Atomic subsumption chains represent only one of many examples of such lemmatisations which preserve transitivity. For the purpose of introducing lemma-isomorphism as an equivalence relation, it suffices to focus on this particular type of lemmatisations, as it captures a frequently occurring pattern in OWL justifications.

5 Diversity of Reason in the NCBO BioPortal Ontologies

5.1 Test Corpus

We performed a survey of equivalence relations in OWL- and OBO-ontologies from the NCBO BioPortal. The BioPortal provides ontologies from various groups from the biomedical domain, including the full set of daily updated OBO Foundry⁵ ontologies, which are built based on common design principles. OBO ontologies use a flat-file format, which can be translated into OWL 2 and were therefore included in the test corpus.

At the time of downloading (January 2012), the BioPortal listed 278 OWL- and OBO-ontologies, of which 241 could be downloaded, merged with their imports, and serialised as OWL/XML. 15 of those ontologies could not be processed in the given time frame of 30 minutes using the selected reasoner, and another 25 did not contain any relevant entailments (direct subsumptions between named classes). For the remaining 201 ontologies, we computed justifications for all entailments with a maximum of 500 justifications per entailment. Self-supporting entailments and self-justifications were excluded from the survey, which led to the discarding of further ontologies.

⁵ <http://obofoundry.org>

The final corpus of justifications consisted of 6,744 justifications from 83 ontologies, covering a broad spectrum of sizes and complexity. Half of the ontologies had less than 1000 named concepts and axioms, with the other half reaching a maximum of 13,959 concepts and 70,015 axioms. Likewise, the expressivity of the corpus ranged from AL to several highly expressive samples in *SRIQ*, which corresponds to the OWL 2 language. A detailed listing of the surveyed ontologies, as well the complete data from the study is available online.⁶

5.2 Implementation

The algorithm for detecting structural similarities between justifications (and therefore, OWL axioms) uses a purely syntactic search for matching subexpressions of OWL axioms. The axioms are first parsed into a tree form, which allows for pairwise comparison of the nodes which represent connectives as well as concept and role names. The implementation uses the OWL API (v3.2.4.) and the Hermit (v1.3.6.) reasoner. Our experiments were performed on a 3GHz Intel Core2 Duo machine with 4GB of RAM allocated to the Java Virtual Machine. The 7051 justifications in the test corpus could be analysed for strict isomorphism in approximately 3 minutes.

5.3 Results of the Survey

Isomorphic Justifications Strict isomorphism applied to all justifications of the individual ontologies drastically reduces the number of regular justifications from an average of 81.3 ($\sigma = 185.5$) justifications per ontology to 10.5 ($\sigma = 18.0$) templates for equivalent justifications. The mean number of justifications per template is 7.7 ($\sigma = 41.7$), which means that in each ontology nearly 8 justifications have an identical structure. This effect is highly visible in the *Orphanet Ontology of Rare Diseases*, where the a single template covers 901 (of 1139) justifications for *distinct* entailments. These justifications are all of the type $\{gen_x \sqsubseteq (\exists geneOf.pat_y), Domain(geneOf, gen_id_1)\} \models gen_x \sqsubseteq gen_id_1$ where x and y denote identifiers used in the ontology.

Likewise, in the *Cognitive Atlas* ontology, all 401 justifications for a *single* entailment are reduced to only one template. Intuitively, it seems that we can observe a much stronger reduction on a large set of justifications than those ontologies that produce only 1 or 2 justifications. And indeed, there exists a strong correlation between the number of justifications of an ontology and the amount of reduction (Spearman's $\rho = 0.78$). However, even some of the larger sets of justifications are only reduced by a small proportion. The 95 justifications of the *Gene Regulation Ontology* are represented by 61 distinct templates, which indicates a fairly diverse corpus.

When applied across all justifications from the corpus, strict isomorphism reduces the corpus from 6,744 justifications to only 614 templates, a reduction

⁶ <http://owl.cs.manchester.ac.uk/research/publications/supporting-material/just-iso-dl2012>

to only 9.1% of the original corpus. On average, 11 justifications share the same template, with the most frequent template occurring 1,603 times across 18 different ontologies; coincidentally, this template is of the same form as the Orphanet Ontology described above. The most prevalent template in the corpus, based on its occurrence in 37 distinct ontologies, is an atomic subsumption chain with 2 axioms.

Subexpression-Isomorphism The reduction from strict isomorphism to s-isomorphism is clearly less drastic than the difference between the main pool and the non-isomorphic pool. The justifications of the 83 ontologies are reduced from an average of 81.3 justifications to 8.8 templates ($\sigma = 13.1$), which is a reduction by 1.7 templates compared to strict isomorphism. An average of 9.2 justifications ($\sigma = 46.6$) in an ontology share the same template.

Surprisingly, the majority of ontologies (67) does not show any difference between strict isomorphism and s-isomorphism. Only 2 ontologies, the *Lipid Ontology* and *Bleeding History Phenotype*, are significantly affected by s-isomorphism, with a reduction from 118 to 13 templates (11.0%) and 32 to 14 templates (43.8%), respectively. Recall that two justifications are s-isomorphic, if their different complex subexpressions can be mapped to an atomic variable name. The significant reduction therefore suggests that in these two ontologies complex expressions are frequently used in the same way as atomic concepts.

Closer inspection reveals, however, that a large number of justifications in the Lipid Ontology consist of one axiom of the form $A \equiv B \sqcap x$, with the entailment being $A \sqsubseteq B$. Here, x represents a number of complex expressions of varying nesting depth. While s-isomorphism captures these types of justifications as equivalent, the actual reason for their similarity lies in their identical *cores*, with the remainder x being a *superfluous* part (with respect to the given entailment).

Across the entire corpus, the number of justifications is reduced from 6,744 to 456 templates (6.8% of the corpus), which is a further reduction compared to strict isomorphism (614 templates). The most frequent templates in terms of number of justifications and prevalence across all ontologies are the same as for strict isomorphism, with numbers only differing slightly.

Lemma-Isomorphism As with s-isomorphism, the effects of l-isomorphism were not as significant as the first reduction through strict isomorphism. The justifications were further reduced to an average of 7.4 templates per ontology ($\sigma = 11.4$), with 11 justifications per template ($\sigma = 51.5$). Still, 35 of the 83 ontologies show at least a minor difference between s-isomorphism and l-isomorphism, which indicates that they contain at least 1 atomic subsumption chain.

L-isomorphism reduces the 106 justifications generated for the *Cereal Plant Gross Anatomy* ontology to only 14 templates, compared to 29 templates for s-isomorphism. This shows that, while not very frequent, there are indeed ontologies in which justifications with subsumption chains of differing lengths occur.

In the *Plant Ontology*, l-isomorphism reduces the 74 justifications to 12 templates, with one template capturing 32 justifications of varying sizes. These justifications contain atomic subsumption chains ranging from 2 to 6 atomic subsumption axioms, which can all be reduced to a single axiom (namely the entailment of the subsumption chain) in the lemmatised version of the justification.

Across the corpus, l-isomorphism reduces the 6,744 justifications to a mere 384 templates, which is an overall reduction of 94.3%. The effect of lemma-isomorphism is visible when we look at the most prevalent justification, an atomic subsumption chain of size 2, which occurs in 44 (compared to previously 37) ontologies. This chain represents all 701 atomic subsumption chains of differing sizes that can be found in the corpus.

5.4 Counting Distinct Reasons

For the purpose of detecting how many distinct *types* of justifications there are for a given set of entailments, we have seen that it is crucial to focus not on the *material* form of a justification, but rather on the justification templates in an ontology. By only considering the abstract template of a set of justifications, we can represent the reasoning that underlies not only one, but a whole class of justifications in the ontology.

Surprisingly, while the newly introduced equivalence relations, s-isomorphism and l-isomorphism, could be shown to capture some of the structural similarities in the surveyed corpus, a large number of justifications were indeed strictly isomorphic. We can argue, however, that even a small reduction can be beneficial when presenting multiple justifications to OWL ontology developers for the purpose of explaining entailments, as it prevents repetitive actions and gives a higher-level view of the set of justifications. This is made obvious in the example of the Plant Ontology, where a large number of justifications that differed only in the length of their atomic subsumption chains, could be captured by a single template.

Another aspect to take into account when dealing with OWL justifications is the *superfluosity* of expressions, as we have seen in the case of the Lipid Ontology. Two justifications may be nearly identical and only differ in expressions that are not necessary for the entailment to hold; these superfluous parts would prevent them from being isomorphic with respect to any of the above definitions, but it is clear that their *form* is the same. This situation describes one of several types of justification *masking* [8]. In order to prevent distortion caused by masking effects, we may want to focus on a type of justifications which is minimal with respect to its subexpressions.

A laconic justification [6] is a justification which does not contain any superfluous parts, with every subexpression being as *weak* as possible. A comparison of the equivalence relations between the regular justifications for the above set of ontologies and their laconic versions as part of future work will allow us to gain further insight into the effect of superfluosity on the diversity of justifications.

6 Conclusions and Future Work

In this paper, we introduced new types of equivalence relations between OWL justifications, subexpression-isomorphism and lemma-isomorphism. We demonstrated how a seemingly diverse corpus of justifications from the NCBO BioPortal could be reduced by over 90% to a much smaller set of non-isomorphic justifications. We have found that, surprisingly, most justifications are in fact strictly isomorphic, with only a few ontologies being affected by the other equivalence relations.

Future work will involve exploring further notions of obvious proof steps in order to extend lemma-isomorphism beyond atomic subsumption chains. We will also consider the issue of overlapping chains, i.e. subsumption chains which lead to non-summarising lemmas. Since the present paper demonstrates the effects of the different equivalence relations on only a subset of the justifications from the BioPortal corpus, we are planning a survey of the entire set of justifications, taking into account the differences between laconic and non-laconic justifications. Finally, we aim to obtain more detailed knowledge about the application of ontology design patterns in the surveyed ontologies, which will allow us to investigate the relations between these design patterns and the form and frequency of justification templates.

References

1. Baader, F., Küsters, R., Borgida, A., McGuinness, D.: Matching in description logics. *Journal of Logic and Computation* 9(3), 411–447 (1999)
2. Baader, F., Narendran, P.: Unification of concept terms in description logics. In: *Proc. of ECAI-98*. pp. 331–335 (1998)
3. Brandt, S.: Implementing matching in $\mathcal{AL}\mathcal{E}$ —first results. In: *Proc. of DL-03* (2003)
4. Davis, M.: Obvious logical inferences. In: *Proc. of IJCAI-81*. pp. 530–531 (1981)
5. Horridge, M., Bail, S., Parsia, B., Sattler, U.: The cognitive complexity of OWL justifications. In: *Proc. of ISWC-11*. pp. 241–256 (2011)
6. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: *Proc. of ISWC-08*. pp. 323–338 (2008)
7. Horridge, M., Parsia, B., Sattler, U.: Lemmas for justifications in OWL. In: *Proc. of DL 2009* (2009)
8. Horridge, M., Parsia, B., Sattler, U.: Justification masking in OWL. In: *Proc. of DL 2010* (2010)
9. Horridge, M., Parsia, B., Sattler, U.: Justification oriented proofs in OWL. In: *Proc. of ISWC-10*. pp. 354–369 (2010)
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SRITQ*. In: *Proc. of KR-06*. pp. 57–67 (2006)
11. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: *Proc. of WWW-05*. pp. 633–640 (2005)
12. P.N., Johnson-Laird: Mental models in cognitive science. *Cognitive Science* 4(1), 71–115 (1980)
13. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: *Proc. of IJCAI-03*. pp. 355–362 (2003)

Inconsistency-Tolerant Conjunctive Query Answering for Simple Ontologies

Meghyn Bienvenu

LRI - CNRS & Université Paris-Sud, France
www.lri.fr/~meghyn/ meghyn@lri.fr

1 Introduction

In recent years, there has been growing interest in using description logic (DL) ontologies to query instance data. An important issue which arises in this setting is how to handle the case in which the data (ABox) is inconsistent with the ontology (TBox). Ideally, one would like to restore consistency by identifying and correcting the errors in the data (using e.g. techniques for debugging or revising DL knowledge bases, cf. [13]). However, such an approach requires the ability to modify the data and the necessary domain knowledge to determine which part of the data is erroneous. When these conditions are not met (e.g. in information integration applications), an alternative is to adopt an inconsistency-tolerant semantics in order to obtain reasonable answers despite the inconsistencies.

The related problem of querying databases which violate integrity constraints has long been studied in the database community (cf. [1] and the survey [6]), under the name of *consistent query answering*. The semantics is based upon the notion of a repair, which is a database which satisfies the integrity constraints and is as similar as possible to the original database. Consistent query answering corresponds to evaluating the query in each of the repairs, and then intersecting the results. This semantics is easily adapted to the setting of ontology-based data access, by defining repairs as the inclusion-maximal subsets of the data which are consistent with the ontology.

Consistent query answering for the *DL-Lite* family of lightweight DLs was investigated in [10, 11]. The obtained complexity results are rather disheartening: the problem was shown in [10] to be co-NP-hard in data complexity, even for instance queries; this contrasts sharply with the very low AC_0 data complexity for (plain) conjunctive query answering in *DL-Lite*. Similarly discouraging results were recently obtained in [14] for another prominent lightweight DL \mathcal{EL}_\perp [3]. In fact, we will see in Example 1 that if we consider conjunctive queries, only a single concept disjointness axiom is required to obtain co-NP-hard data complexity.

In the database community, negative complexity results spurred a line of research [8, 9, 15] aimed at identifying cases where consistent query answering is feasible, and in particular, can be done using first-order query rewriting techniques. The idea is to use targeted polynomial-time procedures whenever possible, and to reserve generic methods with worst-case exponential behavior for difficult cases (see [9] for some experimental results supporting such an approach).

A similar investigation for *DL-Lite* ontologies was initiated in [4], where general conditions were identified for proving either first-order expressibility or coNP-hardness of consistent query answering for a given TBox and instance query.

The main objective of the present work is to gain a better understanding of what makes consistent conjunctive query answering in the presence of ontologies so difficult. To this end, we conduct a fine-grained complexity analysis which aims to characterize the complexity of consistent query answering based on the properties of the ontology and the conjunctive query. We focus on simple ontologies, consisting of class subsumption ($A_1 \sqsubseteq A_2$) and class disjointness ($A_1 \sqsubseteq \neg A_2$) axioms, since the problem is already far from trivial for this case. We identify the number of quantified variables in the query as an important factor in determining the complexity of consistent query answering. Specifically, we show that consistent query answering is always first-order expressible for conjunctive queries with at most one quantified variable; the problem has polynomial data complexity (but is not necessarily first-order expressible) when there are two quantified variables; and it may become coNP-hard starting from three quantified variables. For queries having at most two quantified variables, we further identify a necessary and sufficient condition for first-order expressibility.

To obtain positive results for arbitrary conjunctive queries, we propose a novel inconsistency-tolerant semantics which is a sound approximation of the consistent query answering semantics (and a finer approximation than the approximate semantics proposed in [10]). We show that under this semantics, first-order expressibility of consistent query answering is guaranteed for all conjunctive queries. Finally, in order to treat more expressive ontologies, and to demonstrate the applicability of our techniques, we show how our positive results can be extended to handle *DL-Lite_{core}* ontologies without inverse roles.

Full proofs can be found in a long version available on the author's website.

2 Preliminaries

Syntax. All the ontology languages considered in this paper are fragments of *DL-Lite_{core}* [5, 2]. We recall that *DL-Lite_{core}* knowledge bases (KBs) are built up from a set \mathbf{N}_I of *individuals*, a set \mathbf{N}_C of *atomic concepts*, and a set \mathbf{N}_R of *atomic roles*. Complex concept and role expressions are constructed as follows:

$$B \rightarrow A \mid \exists P \quad C \rightarrow B \mid \neg B \quad P \rightarrow R \mid R^-$$

where $A \in \mathbf{N}_C$ and $R \in \mathbf{N}_R$. A *TBox* is a finite set of *inclusions* of the form $B \sqsubseteq C$ (B, C as above). An *ABox* is a finite set of (*ABox*) *assertions* of the form $A(a)$ ($A \in \mathbf{N}_C$) or $R(a, b)$ ($R \in \mathbf{N}_R$), where $a, b \in \mathbf{N}_I$. We use $\text{Ind}(\mathcal{A})$ to denote the set of individuals in \mathcal{A} . A KB consists of a TBox and an ABox.

Semantics An *interpretation* is $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ maps each $a \in \mathbf{N}_I$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each $A \in \mathbf{N}_C$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each $P \in \mathbf{N}_R$ to $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is straightforwardly extended to general concepts and roles, e.g. $(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$ and $(\exists S)^{\mathcal{I}} = \{c \mid \exists d : (c, d) \in S^{\mathcal{I}}\}$.

\mathcal{I} satisfies $G \sqsubseteq H$ if $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$; it satisfies $A(a)$ (resp. $P(a, b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$). We write $\mathcal{I} \models \alpha$ if \mathcal{I} satisfies inclusion/assertion α . An interpretation \mathcal{I} is a *model* of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if \mathcal{I} satisfies all inclusions in \mathcal{T} and assertions in \mathcal{A} . We say a KB \mathcal{K} is *consistent* if it has a model, and that \mathcal{K} *entails* an inclusion/assertion α , written $\mathcal{K} \models \alpha$, if every model of \mathcal{K} is a model of α .

We say that a set of concepts $\{C_1, \dots, C_n\}$ is consistent w.r.t. a TBox \mathcal{T} if there is a model \mathcal{I} of \mathcal{T} and an element $e \in \Delta^{\mathcal{I}}$ such that $e \in C_i$ for every $1 \leq i \leq n$. Entailment of a concept from a set of concepts is defined in the obvious way: $\mathcal{T} \models S \sqsubseteq D$ if and only if for every model \mathcal{I} of \mathcal{T} , we have $\bigcap_{C \in S} C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Queries A (*first-order*) *query* is a formula of first-order logic with equality, whose atoms are of the form $A(t)$ ($A \in \mathbf{N}_C$), $R(t, t')$ ($R \in \mathbf{N}_R$), or $t = t'$ with t, t' *terms*, i.e., variables or individuals. *Conjunctive queries* (CQs) have the form $\exists \mathbf{y} \psi$, where \mathbf{y} denotes a tuple of variables, and ψ is a conjunction of atoms of the forms $A(t)$ or $R(t, t')$. *Instance queries* are queries consisting of a single atom with no variables (i.e. ABox assertions). Free variables in queries are called *answer variables*, whereas bound variables are called *quantified variables*. We use $\text{terms}(q)$ to denote the set of terms appearing in a query q .

A *Boolean query* is a query with no answer variables. For a Boolean query q , we write $\mathcal{I} \models q$ when q holds in the interpretation \mathcal{I} , and $\mathcal{K} \models q$ when $\mathcal{I} \models q$ for all models \mathcal{I} of \mathcal{K} . For a non-Boolean query q with answer variables v_1, \dots, v_k , a tuple of individuals (a_1, \dots, a_k) is said to be a certain answer for q w.r.t. \mathcal{K} just in the case that $\mathcal{K} \models q[a_1, \dots, a_k]$, where $q[a_1, \dots, a_k]$ is the Boolean query obtained by replacing each v_i by a_i . Thus, conjunctive query answering is straightforwardly reduced to entailment of Boolean CQs.

First-order rewritability Calvanese et al. [5] proved that for every *DL-Lite_{core}* TBox \mathcal{T} and CQ q , there exists a first-order query q' such that for every ABox \mathcal{A} and tuple \mathbf{a} : $\mathcal{T}, \mathcal{A} \models q[\mathbf{a}] \Leftrightarrow \mathcal{I}_{\mathcal{A}} \models q'[\mathbf{a}]$, where $\mathcal{I}_{\mathcal{A}}$ denotes the interpretation with domain $\text{Ind}(\mathcal{A})$ that makes true precisely the assertions in \mathcal{A} .

3 Consistent Query Answering for Description Logics

In this section, we formally recall the consistent query answering semantics, present some simple examples which illustrate the difficulty of the problem, and introduce the main problem which will be studied in this paper. For readability, we will formulate our definitions and results in terms of Boolean CQs, but they can be straightforwardly extended to general CQs.

The key notion underlying consistent query answering semantics is that of a repair of an ABox \mathcal{A} , which is an ABox which is consistent with the TBox and as similar as possible to \mathcal{A} . In this paper, we follow common practice and use subset inclusion to compare ABoxes.

Definition 1. A repair of a DL ABox \mathcal{A} w.r.t. a TBox \mathcal{T} is an inclusion-maximal subset \mathcal{B} of \mathcal{A} consistent with \mathcal{T} . We use $\text{Rep}_{\mathcal{T}}(\mathcal{A})$ to denote the set of repairs of \mathcal{A} w.r.t. \mathcal{T} .

Consistent query answering can be seen as performing standard query answering on each of the repairs and intersecting the answers. For Boolean queries, the formal definition is as follows:

Definition 2. A query q is said to be consistently entailed from a DL KB $(\mathcal{T}, \mathcal{A})$, written $\mathcal{T}, \mathcal{A} \models_{\text{cons}} q$, if $\mathcal{T}, \mathcal{B} \models q$ for every repair $\mathcal{B} \in \text{Rep}_{\mathcal{T}}(\mathcal{A})$.

Just as with standard query entailment, we can ask whether consistent query entailment can be tested by rewriting the query and evaluating it over the data.

Definition 3. A first-order query q' is a consistent rewriting of a Boolean query q w.r.t. a TBox \mathcal{T} if for every ABox \mathcal{A} , we have $\mathcal{T}, \mathcal{A} \models_{\text{cons}} q$ iff $\mathcal{I}_{\mathcal{A}} \models q'$.

As mentioned in Section 1, consistent query answering in $DL\text{-}Lite_{\text{core}}$ is co-NP-hard in data complexity, even for instance queries [10], which means in particular that consistent rewritings need not exist. All known reductions make crucial use of inverse roles, and indeed, we will show in Section 7 that consistent instance checking is first-order expressible for $DL\text{-}Lite_{\text{core}}$ ontologies without inverse. However, in the case of conjunctive queries, the absence of inverses does not guarantee tractability. Indeed, the next example shows that only a single concept disjointness axiom can yield coNP-hardness.

Example 1. We use a variant of UNSAT, called 2+2UNSAT, proved coNP-hard in [7], in which each clause has 2 positive and 2 negative literals, where literals involve either regular variables or the truth constants **true** and **false**. Consider an instance $\varphi = c_1 \wedge \dots \wedge c_m$ of 2+2-UNSAT over v_1, \dots, v_k , **true**, and **false**. Let $\mathcal{T} = \{T \sqsubseteq \neg F\}$, and define \mathcal{A} as follows:

$$\begin{aligned} & \{ P_1(c_i, u), P_2(c_i, x), N_1(c_i, y), N_2(c_i, z) \mid c_i = u \vee x \vee \neg y \vee \neg z, 1 \leq i \leq m \} \\ & \cup \{ T(v_j), F(v_j) \mid 1 \leq j \leq k \} \cup \{ T(\mathbf{true}), F(\mathbf{false}) \} \end{aligned}$$

Then one can show that φ is unsatisfiable just in the case that $(\mathcal{T}, \mathcal{A})$ consistently entails the following query:

$$\exists x y_1 \dots y_4 P_1(x, y_1) \wedge F(y_1) \wedge P_2(x, y_2) \wedge F(y_2) \wedge N_1(x, y_3) \wedge T(y_3) \wedge N_2(x, y_4) \wedge T(y_4)$$

Essentially, $T \sqsubseteq \neg F$ forces the choice of a truth value for each variable, so the repairs of \mathcal{A} correspond exactly to the set of valuations. Importantly, there is only one way to avoid satisfying a 2+2-clause: the first two variables must be assigned false and the last two variables must be assigned true. The existence of such a configuration is checked by q .

We remark that the query in the preceding reduction does not have a particularly complicated structure (in particular, it is tree-shaped). Its only notable property is that it has several quantified variables.

In this paper, we aim to gain a better understanding of what makes consistent conjunctive query answering so difficult (and conversely, what can make it easy). To this end, we will consider the following decision problem:

CONS_{ENT}(q, \mathcal{T}): Is \mathcal{A} such that $\mathcal{T}, \mathcal{A} \models_{\text{cons}} q$?

and we will try to characterize its complexity in terms of the properties of the pair (q, \mathcal{T}) . We will in particular investigate the impact of limiting the number of quantified variables in the query q .

In the next three sections, we focus on *simple ontologies*, consisting of inclusions of the forms $A_1 \sqsubseteq A_2$ and $A_1 \sqsubseteq \neg A_2$ where $A_1, A_2 \in \mathbf{N}_C$. As Example 1 demonstrates, the problem is already non-trivial in this case. All obtained lower bounds transfer to richer ontologies, and we will show in Section 7 that positive results can also be extended to *DL-Lite_{core}* ontologies without inverse roles.

4 Tractability for Queries with At Most Two Quantified Variables

In this section, we investigate the complexity of consistent query answering in the presence of simple ontologies for CQs having at most two quantified variables. We show this problem has tractable data complexity, and we provide necessary and sufficient conditions for FO-expressibility.

We begin with queries with at most one quantified variable, showing that a consistent rewriting always exists.

Theorem 1. *Let \mathcal{T} be a simple ontology, and let q be a Boolean CQ with at most one quantified variable. Then $\text{CONS}_{\text{ENT}}(q, \mathcal{T})$ is first-order expressible.*

Proof (Sketch). We show how to construct the desired consistent rewriting of q in the case where q has a single quantified variable x . First, for each $t \in \text{terms}(q)$, we set $C_t = \{A \mid A(t) \in q\}$, and we let Σ_t be the set of all $S \subseteq \mathbf{N}_C$ such that every maximal subset $U \subseteq S$ consistent with \mathcal{T} is such that $\mathcal{T} \models U \sqsubseteq C_t$. Intuitively, Σ_t defines the possible circumstances under which the conjunction of concepts in C_t is consistently entailed. We can express this condition with the first-order formula ψ_t :

$$\psi_t = \bigvee_{S \in \Sigma_t} \left(\bigwedge_{A \in S} A(t) \wedge \bigwedge_{A \in \mathbf{N}_C \setminus S} \neg A(t) \right)$$

Now using the ψ_t , we construct q' :

$$q' = \exists x \bigwedge_{R(t, t') \in q} R(t, t') \wedge \bigwedge_{t \in \text{terms}(q)} \psi_t$$

It can be shown that q' is indeed a consistent rewriting of q w.r.t. \mathcal{T} . To see why this is so, it is helpful to remark that the repairs of $(\mathcal{T}, \mathcal{A})$ contain precisely the role assertions in \mathcal{A} , together with a maximal subset of concept assertions consistent with \mathcal{T} for each individual.

The next example shows that Theorem 1 cannot be extended to the class of queries with two quantified variables.

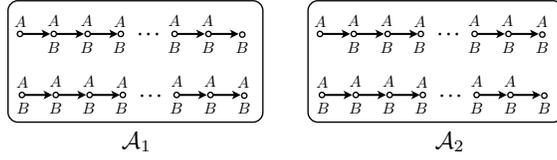


Fig. 1: ABoxes for Example 2. Arrows indicate the role R , and each of the four R -chains has length exceeding 2^k .

Example 2. Consider $q = \exists xy A(x) \wedge R(x, y) \wedge B(y)$ and $\mathcal{T} = \{A \sqsubseteq \neg B\}$. Suppose for a contradiction that q' is a consistent rewriting of q w.r.t. \mathcal{T} , and let k be the quantifier rank of q' . In Fig. 1, we give two ABoxes \mathcal{A}_1 and \mathcal{A}_2 , each consisting of two R -chains of length $> 2^k$. It can be verified that q is consistently entailed from $\mathcal{T}, \mathcal{A}_1$. This is because in every repair, the upper chain will have A at one end, B at the other, and either an A or B at all interior points; every such configuration makes q true somewhere along the chain. On the other hand, we can construct a repair for $\mathcal{T}, \mathcal{A}_2$ which does not entail q by always preferring A on the top chain and B on the bottom chain. It follows that the interpretation $\mathcal{I}_{\mathcal{A}_1}$ satisfies q' , whereas $\mathcal{I}_{\mathcal{A}_2}$ does not. However, one can show using standard tools from finite model theory (cf. Ch. 3-4 of [12]) that no formula of quantifier rank k can distinguish $\mathcal{I}_{\mathcal{A}_1}$ and $\mathcal{I}_{\mathcal{A}_2}$, yielding the desired contradiction.

We can generalize the preceding example to obtain sufficient conditions for the inexistence of a consistent rewriting.

Theorem 2. *Let \mathcal{T} be a simple ontology, and let q be a Boolean CQ with two quantified variables x, y . Assume that there do not exist CQs q_1 and q_2 , each with less than two quantified variables, such that $q \equiv q_1 \wedge q_2$. Denote by C_x (resp. C_y) the set of concepts A such that $A(x) \in q$ (resp. $A(y) \in q$). Then $\text{CONSENT}(q, \mathcal{T})$ is not first-order expressible if there exists $S \subseteq \mathbf{N}_C$ such that:*

- for $v \in \{x, y\}$, there is a maximal subset $D_v \subseteq S$ consistent with \mathcal{T} s.t. $\mathcal{T} \not\models D_v \sqsubseteq C_v$
- for every maximal subset $D \subseteq S$ consistent with \mathcal{T} , either $\mathcal{T} \models D \sqsubseteq C_x$ or $\mathcal{T} \models D \sqsubseteq C_y$

Proof (Sketch). The proof generalizes the argument outlined in Example 2. Instead of having a single role connecting successive elements in the chains, we establish the required relational structure for each pair of successive points. We then substitute the set D_y for A , the set D_x for B , and the set S for $\{A, B\}$. The properties of S ensure that if S is asserted at some individual, then we can block the satisfaction of C_x using D_y , and we can block C_y using D_x , but we can never simultaneously block both C_x and C_y . The assumption that q cannot be rewritten as a conjunction of queries with less than two quantified variables is used in the proof of $\mathcal{T}, \mathcal{A}_2 \not\models_{\text{cons}} q$ to show that the only possible matches of q involve successive chain elements (and not constants from the query). To show $\mathcal{I}_{\mathcal{A}_1}$ and $\mathcal{I}_{\mathcal{A}_2}$ cannot be distinguished, we use Ehrenfeucht-Fraïssé games,

rather than Hanf locality, since the latter is inapplicable when there is a role atom containing a constant and a quantified variable.

The following theorem shows that whenever the conditions of Theorem 2 are not met, a consistent rewriting exists.

Theorem 3. *Let \mathcal{T} be a simple ontology, and let q be a Boolean CQ with two quantified variables x, y . Then $\text{CONSENT}(q, \mathcal{T})$ is first-order expressible if q is equivalent to a CQ with at most one quantified variable, or if there is no set S satisfying the conditions of Theorem 2.*

Proof (Sketch). When q is equivalent to a query q' with at most one quantified variable, then Theorem 1 yields a consistent rewriting of q' , and hence of q . Thus, the interesting case is when there is no such equivalent query, nor any set S satisfying the conditions of Theorem 2. Intuitively, the inexistence of such a set S ensures that if at some individual, one can block C_x , and one can block C_y , then it is possible to simultaneously block C_x and C_y (compare this to Example 2 in which blocking A causes B to hold, and vice-versa). This property is key, as it allows different potential query matches to be treated independently.

Together, Theorems 2 and 3 provide a necessary and sufficient condition for the existence of a consistent rewriting. We now reconsider \mathcal{T} and q from Example 2 and outline a polynomial-time method for solving $\text{CONSENT}(q, \mathcal{T})$.

Example 3. Suppose we have an ABox \mathcal{A} , and we wish to decide if $\mathcal{T}, \mathcal{A} \models_{\text{cons}} q$, for $\mathcal{T} = \{A \sqsubseteq \neg B\}$ and $q = \exists xy A(x) \wedge R(x, y) \wedge B(y)$. The basic idea is to try to construct a repair which does not entail q . We start by iteratively applying the following rules until neither rule is applicable: (1) if $R(a, b), A(a), B(a), B(b) \in \mathcal{A}$ but $A(b) \notin \mathcal{A}$, then delete $A(a)$ from \mathcal{A} , and (2) if $R(a, b), A(a), A(b), B(b) \in \mathcal{A}$ but $B(a) \notin \mathcal{A}$, then delete $B(b)$. Note that since the size of \mathcal{A} decreases with every rule application, we will stop after a polynomial number of iterations. Once finished, we check whether there are a, b such that $A(a), R(a, b), B(b) \in \mathcal{A}$, $B(a) \notin \mathcal{A}$, and $A(b) \notin \mathcal{A}$. If so, we return ‘yes’ (to indicate $\mathcal{T}, \mathcal{A} \models_{\text{cons}} q$), and otherwise, we output ‘no’ (for $\mathcal{T}, \mathcal{A} \not\models_{\text{cons}} q$). Note that in the latter case, for all pairs a, b with $A(a), R(a, b), B(b) \in \mathcal{A}$, we have both $B(a)$ and $A(b)$. Thus, we can choose to always keep A , thereby blocking all remaining potential matches.

By carefully generalizing the ideas outlined in Example 3, we obtain a tractability result which covers all queries having at most two quantified variables.

Theorem 4. *Let \mathcal{T} be a simple ontology, and let q be a CQ with at most 2 quantified variables. Then $\text{CONSENT}(q, \mathcal{T})$ is polynomial in data complexity.*

5 An Improved coNP Lower Bound

The objective of this section is to show that the tractability result we obtained for queries with at most two quantified variables cannot be extended further

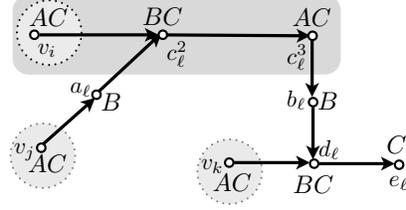


Fig. 2: Abox \mathcal{A}_{c_ℓ} for clause $c_\ell = \neg v_i \vee \neg v_j \vee \neg v_k$

to the class of conjunctive queries with three quantified variables. We will do this by establishing coNP-hardness for a specific conjunctive query with three quantified variables, thereby improving the lower bound sketched in Example 1. Specifically, we will reduce 3SAT to $\text{CONSENT}(q, \mathcal{T})$ where:

$$\mathcal{T} = \{A \sqsubseteq \neg B, A \sqsubseteq \neg C, B \sqsubseteq \neg C\}$$

$$q = \exists x, y, z A(x) \wedge R(x, y) \wedge B(y) \wedge R(y, z) \wedge C(z).$$

The first component of the reduction is a mechanism for choosing truth values for the variables. For this, we create an ABox $\mathcal{A}_{v_i} = \{A(v_i), C(v_i)\}$ for each variable v_i . It is easy to see that there are two repairs for \mathcal{A}_{v_i} w.r.t. \mathcal{T} : $\{A(v_i)\}$ and $\{C(v_i)\}$. We will interpret the choice of $A(v_i)$ as assigning true to v_i , and the presence of $C(v_i)$ to mean that v_i is false.

Next we need some way of verifying whether a clause is satisfied by the valuation associated with a repair of $\cup_i \mathcal{A}_{v_i}$. To this end, we create an ABox \mathcal{A}_{c_ℓ} for each clause c_ℓ ; the ABox \mathcal{A}_φ encoding φ will then simply be the union of the ABoxes \mathcal{A}_{v_i} and \mathcal{A}_{c_ℓ} . The precise definition of the ABox \mathcal{A}_{c_ℓ} is a bit delicate and depends on the polarity of the literals in c_ℓ . Figure 2 presents a pictorial representation of \mathcal{A}_{c_ℓ} for the case where $c_\ell = \neg v_i \vee \neg v_j \vee \neg v_k$ (the ABoxes \mathcal{A}_{v_i} , \mathcal{A}_{v_j} , and \mathcal{A}_{v_k} are also displayed).

Let us now see how the ABox \mathcal{A}_{c_ℓ} pictured in Fig. 2 can be used to test the satisfaction of c_ℓ . First suppose that we have a repair \mathcal{B} of \mathcal{A}_φ which contains $A(v_i)$, $A(v_j)$, and $A(v_k)$, i.e. the valuation associated with the repair does not satisfy c_ℓ . We claim that this implies that q holds. Suppose for a contradiction that q is not entailed from \mathcal{T}, \mathcal{B} . We first note that by maximality of repairs, \mathcal{B} must contain all of the assertions $A(v_j)$, $R(v_j, a_\ell)$, $B(a_\ell)$, and $R(a_\ell, c_\ell^2)$. It follows that including $C(c_\ell^2)$ in \mathcal{B} would cause q to hold, which means we must choose to include $B(c_\ell^2)$ instead. Using similar reasoning, we can see that in order to avoid satisfying q , we must have $C(d_\ell)$ in \mathcal{B} rather than $B(d_\ell)$, which in turn forces us to select $C(c_\ell^3)$ to block $A(c_\ell^3)$. However, this is a contradiction, since we have identified a match for q in \mathcal{B} with $x = v_i, y = c_\ell^2, z = c_\ell^3$. The above argument (once extended to the other possible forms of \mathcal{A}_{c_ℓ}) is the key to showing that the unsatisfiability of φ implies $\mathcal{T}, \mathcal{A}_\varphi \models q$.

Conversely, it can be proven that if one of c_ℓ 's literals is made true by the valuation, then it is possible to repair \mathcal{A}_{c_ℓ} in such a way that a match for q is avoided. For example, consider again \mathcal{A}_{c_ℓ} from Figure 2, and suppose that

the second literal v_j is satisfied. It follows that $C(v_j) \in \mathcal{B}$, hence $A(v_j) \notin \mathcal{B}$, which means we can keep $C(c_\ell^2)$ rather than $B(c_\ell^2)$, thereby blocking the match at $(v_i, c_\ell^2, c_\ell^3)$. By showing this property holds for the different forms of \mathcal{A}_{c_ℓ} , and by further arguing that we can combine “ q -avoiding” repairs of the \mathcal{A}_{c_ℓ} without inducing a match for q , we can prove that the satisfiability of φ implies $\mathcal{T}, \mathcal{A}_\varphi \not\models q$. We thus have:

Theorem 5. $\text{CONSENT}(q, \mathcal{T})$ is coNP-hard in data complexity for $\mathcal{T} = \{A \sqsubseteq \neg B, A \sqsubseteq \neg C, B \sqsubseteq \neg C\}$ and $q = \exists x, y, z A(x) \wedge R(x, y) \wedge B(y) \wedge R(y, z) \wedge C(z)$.

6 Tractability through Approximation

The positive results from Section 4 give us a polynomial algorithm for consistent query answering in the presence of simple ontologies, but only for CQs with at most two quantified variables. In order to be able to handle all queries, we explore in this section alternative inconsistency-tolerant semantics which are sound approximations of the consistent query answering semantics.

One option is to adopt the IAR semantics from [10]. We recall that this semantics (denoted by \models_{IAR}) can be seen as evaluating queries against the ABox corresponding to the *intersection of the repairs*. Conjunctive query answering under IAR semantics was shown in [11] tractable for general CQs in the presence of DL-Lite ontologies (and *a fortiori* simple ontologies) using query rewriting.

To obtain a finer approximation of the consistent query answering semantics, we propose a new inconsistency-tolerant semantics which corresponds to closing repairs with respect to the TBox before intersecting them. In the following definition, we use $cl_{\mathcal{T}}(\mathcal{B})$ to denote the set of assertions entailed from \mathcal{T}, \mathcal{B} .

Definition 4. A Boolean query q is said to be entailed from $(\mathcal{T}, \mathcal{A})$ under ICR semantics (“*intersection of closed repairs*”), written $\mathcal{T}, \mathcal{A} \models_{ICR} q$, if $\mathcal{T}, \mathcal{D} \models q$, where $\mathcal{D} = \bigcap_{\mathcal{B} \in \text{Rep}_{\mathcal{T}}(\mathcal{A})} cl_{\mathcal{T}}(\mathcal{B})$.

The following theorem, which is easy to prove, establishes the relationship among the three semantics.

Theorem 6. For every Boolean CQ q and TBox \mathcal{T} :

$$\mathcal{T}, \mathcal{A} \models_{IAR} q \quad \Rightarrow \quad \mathcal{T}, \mathcal{A} \models_{ICR} q \quad \Rightarrow \quad \mathcal{T}, \mathcal{A} \models_{cons} q$$

The reverse implications do not hold.

The next example illustrates the difference between IAR and ICR semantics:

Example 4. Let $\mathcal{T} = \{A \sqsubseteq C, B \sqsubseteq C, A \sqsubseteq \neg B\}$ and $\mathcal{A} = \{A(a), B(a)\}$. Then $C(a)$ is entailed from $(\mathcal{T}, \mathcal{A})$ under ICR semantics, but not under IAR semantics.

Finally, we show that under ICR semantics, we can answer any conjunctive query in polynomial time using query rewriting.

Theorem 7. *Let \mathcal{T} be a simple ontology and q a Boolean CQ. Then there exists a first-order query q' such that for every ABox \mathcal{A} : $\mathcal{T}, \mathcal{A} \models_{ICR} q$ iff $\mathcal{I}_{\mathcal{A}} \models q'$.*

Proof (Sketch). We first compute, using standard techniques, a union of conjunctive queries φ such that for every \mathcal{A} , we have $\mathcal{T}, \mathcal{A} \models q$ if and only if $\mathcal{I}_{\mathcal{A}} \models \varphi$. Next we use Theorem 1 to find a consistent rewriting $\psi_{A(t)}$ of each concept atom $A(t) \in \varphi$, and we let q' be the first-order query obtained by replacing each occurrence of $A(t)$ in φ by $\psi_{A(t)}$. It can be shown that the query q' is such that $\mathcal{T}, \mathcal{A} \models_{ICR} q$ if and only if $\mathcal{I}_{\mathcal{A}} \models q'$.

7 Extension to Inverse-Free $DL-Lite_{core}$

In this section, we show how the techniques we developed for simple ontologies can be used to extend our positive results to $DL-Lite_{core}$ ontologies which do not contain inverse roles (we will use $DL-Lite^{no-}$ to refer to this logic).

Our first result shows that the analogues of Theorems 1 and 4 hold for $DL-Lite^{no-}$ ontologies. The main technical difficulty in adapting the proofs of Theorems 1 and 4 is that role assertions may now be contradicted, which means repairs need not have the same set of role assertions as the original ABox.

Theorem 8. *Consider a $DL-Lite^{no-}$ ontology \mathcal{T} , and a Boolean CQ q with at most two quantified variables. Then $\text{CONSENT}(q, \mathcal{T})$ is polynomial in data complexity, and first-order expressible if there is at most one quantified variable.*

We can also extend the general first-order expressibility result for the new ICR semantics (Theorem 7) to the class of $DL-Lite^{no-}$ ontologies.

Theorem 9. *Let \mathcal{T} be a $DL-Lite^{no-}$ ontology, and let q be a Boolean CQ. Then there exists a first-order query q' such that for every ABox \mathcal{A} : $\mathcal{T}, \mathcal{A} \models_{ICR} q$ if and only if $\mathcal{I}_{\mathcal{A}} \models q'$.*

As noted earlier, consistent query answering in (full) $DL-Lite_{core}$ is coNP-hard in data complexity even for instance queries, which means that neither of the preceding theorems can be extended to the class of $DL-Lite_{core}$ ontologies.

8 Conclusion and Future Work

The detailed complexity analysis we conducted for consistent query answering in the presence of simple ontologies provides further insight into previously obtained negative complexity results [10, 14], by making clear how little is needed to obtain first-order inexpressibility or intractability. Our investigation also yielded some positive results, including the identification of novel tractable cases, such as inverse-free $DL-Lite_{core}$ ontologies coupled with CQs with at most two quantified variables (or coupled with arbitrary CQs, under the new ICR semantics).

There are several natural directions for future work. First, it would be interesting to explore how far we can push our positive results. We expect that

adding Horn inclusions and positive role inclusions should be unproblematic, but role disjointness axioms will be more challenging. In order to handle functional roles, we might try to combine our positive results with those which have been obtained for relational databases under functional dependencies [15]. It would also be interesting to try to build upon the results in this paper in order to obtain a criterion for first-order expressibility (or tractability) which applies to all conjunctive queries, regardless of the number of quantified variables.

Finally, we view the present work as a useful starting point in the development of sound but incomplete consistent query answering algorithms for popular lightweight DLs like (full) *DL-Lite_{core}* and \mathcal{EL}_{\perp} . For example, our results could be extended to identify some CQ-TBox pairs in these richer logics for which consistent query answering is tractable. Another idea is to use the new ICR semantics to lift tractability results for IQs (like those in [4]) to classes of CQs.

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proc. of PODS. pp. 68–79. ACM Press (1999)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: The DL-Lite family and relations. *Journal of Artificial Intelligence Research* 36, 1–69 (2009)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Proc. of IJCAI. pp. 364–369 (2005)
4. Bienvenu, M.: First-order expressibility results for queries over inconsistent DL-Lite knowledge bases. In: Proc. of DL (2011)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning* 39(3), 385–429 (2007)
6. Chomicki, J.: Consistent query answering: Five easy pieces. In: Proc. of ICDT. pp. 1–17 (2007)
7. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation* 4(4), 423–452 (1994)
8. Fuxman, A., Miller, R.J.: First-order query rewriting for inconsistent databases. In: Proc. of ICDT. pp. 337–351 (2005)
9. Grieco, L., Lembo, D., Rosati, R., Ruzzi, M.: Consistent query answering under key and exclusion dependencies: algorithms and experiments. In: Proc. of CIKM. pp. 792–799 (2005)
10. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Proc. of RR. pp. 103–117 (2010)
11. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Query rewriting for inconsistent DL-Lite ontologies. In: Proc. of RR. pp. 155–169 (2011)
12. Libkin, L.: *Elements of Finite Model Theory*. Springer (2004)
13. Nikitina, N., Rudolph, S., Glimm, B.: Reasoning-supported interactive revision of knowledge bases. In: Proc. of IJCAI. pp. 1027–1032 (2011)
14. Rosati, R.: On the complexity of dealing with inconsistency in description logic ontologies. In: Proc. of IJCAI. pp. 1057–1062 (2011)
15. Wijsen, J.: On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In: Proc. of PODS. pp. 179–190 (2010)

Deciding FO-Rewritability in \mathcal{EL}

Meghyn Bienvenu¹ and Carsten Lutz² and Frank Wolter³

¹ LRI - CNRS & Université Paris Sud, France

² Department of Computer Science, University of Bremen, Germany

³ Department of Computer Science, University of Liverpool, UK

Abstract. We consider the problem of deciding, given an instance query $A(x)$, an \mathcal{EL} -TBox \mathcal{T} , and possibly an ABox signature Σ , whether $A(x)$ is FO-rewritable relative to \mathcal{T} and Σ -ABoxes. Our main results are PSPACE-completeness for the case where Σ comprises all symbols and EXPTIME-completeness for the general case. We also show that the problem is in PTIME for classical TBoxes and that every instance query is FO-rewritable into a polynomial-size FO query relative to every (semi)-acyclic TBox (under some mild assumptions on the data).

1 Introduction

Over the last years, query answering over instance data has developed into one of the most prominent problems in description logic (DL) research. Many approaches aim at utilizing relational databases systems (RDBMSs), exploiting their mature technology, advanced optimization techniques, and the general infrastructure that those systems offer. Roughly, RDBMS-based approaches can be classified into *query rewriting approaches*, where the original query and the DL TBox are compiled into an SQL query that is passed to the RDBMS for execution [5], and *combined approaches*, where the consequences of the TBox are materialized in the data in a compact form and some query rewriting is used to ensure correct answers despite the compact representation [12, 11]. This division is by no means strict, as illustrated by the approach presented in [7] which is based on query rewriting, but also has strong similarities with combined approaches.

A fundamental difference between the query rewriting approach and the combined approach is that, in query rewriting, an exponential blowup of the query is often unavoidable [8] while the combined approach typically blows up both query and data only polynomially [12, 11]. It is thus unsurprising that query execution is more efficient in the combined approach than in the query rewriting approach, see the experiments in [11]. Depending on the application, however, there can still be good reasons to use pure query rewriting. *Ease of implementation:* Query rewriting approaches are often easier to implement as they do not involve a data completion phase. When the TBox is sufficiently small so that the exponential blowup of the query is not prohibitive or when only a prototype implementation is aimed at, it may not be worthwhile to implement a full combined approach. *Access limitations:* If the user does not have permission to modify the data in the database, materializing the consequences of the TBox in the data might simply be out of the question. This problem arises notably in information integration applications.

In this paper, we are interested in TBoxes formulated in the description logic \mathcal{EL} , which forms the basis of the OWL EL fragment of OWL 2 and is popular as a basic language for large-scale ontologies. In general, query rewriting approaches are not applicable to \mathcal{EL} because instance query answering in this DL is PTIME-complete regarding data complexity while AC_0 data complexity marks the boundary of DLs for which the pure query rewriting approach can be made work [5]. For example, the query $A(x)$ cannot be answered by an SQL-based RDBMS in the presence of the very simple \mathcal{EL} -TBox $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$, intuitively because \mathcal{T} forces the concept name A to be *propagated unboundedly* along r -chains in the data and thus the rewritten query would have to express transitive closure of r . We say that $A(x)$ is not *FO-rewritable relative to \mathcal{T}* , alluding to the known equivalence of first-order (FO) formulas and SQL queries.

Of course, such an isolated example does not rule out the possibility that *some* \mathcal{EL} -TBoxes, including those that are used in applications, still enjoy FO-rewritability. For example, the query $A(x)$ is FO-rewritable relative to the \mathcal{EL} -TBox $\mathcal{T}' = \{A \sqsubseteq \exists r.A\}$: since the additional instances of A stipulated by \mathcal{T}' are ‘anonymous objects’ (nulls in database parlance) rather than primary data objects, there is no unbounded propagation *through the data* and, in fact, we can simply drop \mathcal{T}' when answering $A(x)$. Inspired by these observations, the aim of this paper is to study FO-rewritability on the level of individual TBoxes, essentially following the non-uniform approach initiated in [15]. In particular, we are interested in deciding, for a given instance query (IQ) $A(x)$ and \mathcal{EL} -TBox \mathcal{T} , whether q is FO-rewritable relative to \mathcal{T} . Sometimes, we additionally allow as a third input an ABox-signature Σ that restricts the symbols which can occur in the data [2, 3].

Our main result is that deciding FO-rewritability of IQs relative to *general* \mathcal{EL} -TBoxes (sets of concept inclusions $C \sqsubseteq D$) is PSPACE-complete when the ABox signature Σ is full (i.e., all symbols are allowed in the ABox) and EXPTIME-complete when Σ is given as an input. For proving these results, we establish some properties that are of independent interest, such as: (1) whenever an IQ is FO-rewritable, then it is FO-rewritable into a union of tree-shaped conjunctive queries; (2) an IQ is FO-rewritable relative to all ABoxes iff it is FO-rewritable relative to *tree-shaped* ABoxes (see Section 3 for a precise formulation). We also study more restricted forms of TBoxes, showing that FO-rewritability of IQs relative to *classical* TBoxes (sets of concept definitions $A \equiv C$ and concept implications $A \sqsubseteq C$ with A atomic, cycles allowed) is in PTIME, even when Σ is part of the input. For *semi-acyclic* TBoxes \mathcal{T} (classical TBoxes without cycles that involve only concept definitions, but potentially with cycles that involve at least one concept inclusion), we observe that every IQ is FO-rewritable relative to \mathcal{T} (for any ABox signature Σ) and that, under the mild assumption that the admitted databases have domain size at least two, even a polynomial-sized rewriting is possible. While it is not our primary aim in this first publication to actually generate FO-rewritings, we note that all our results come with effective procedures for doing this (the rewritings are of triple-exponential size in the worst case).

Although we focus on simple IQs of the form $A(x)$, all results in this paper also apply to instance queries of the form $C(x)$ with C an \mathcal{EL} -concept. The treatment of conjunctive queries (CQs) is left for future work. We also discuss the connection of FO-rewritability in \mathcal{EL} to boundedness in datalog and in the μ -calculus. Proof details are deferred to the long version at <http://www.informatik.uni-bremen.de/~clu/papers/>.

2 Preliminaries

We remind the reader that \mathcal{EL} -concepts are built up from concept names and the concept \top using conjunction $C \sqcap D$ and existential restriction $\exists r.C$. When we speak of a *TBox* without further qualification, we mean a *general TBox*, i.e., a finite set of *concept inclusions (CIs)* $C \sqsubseteq D$. Other forms of TBoxes will be introduced later as needed. An *ABox* is a finite set of *concept assertions* $A(a)$ and *role assertions* $r(a, b)$ where A is a concept name, r a role name, and a, b individual names. We use $\text{Ind}(\mathcal{A})$ to denote the set of all individual names used in \mathcal{A} . It will sometimes be convenient to view an ABox \mathcal{A} as an interpretation $\mathcal{I}_{\mathcal{A}}$, defined in the obvious way (see [15]).

Regarding query languages, we focus on *instance queries (IQ)*, which have the form $A(x)$ with A a concept name and x a variable. We write $\mathcal{T}, \mathcal{A} \models A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ for all models \mathcal{I} of \mathcal{T} and \mathcal{A} and call a a *certain answer* to $A(x)$ given \mathcal{A} and \mathcal{T} . We use $\text{cert}_{\mathcal{T}}(A(x), \mathcal{A})$ to denote the set of all certain answers to $A(x)$ given \mathcal{A} and \mathcal{T} . To define FO-rewritability, we require first-order queries (FOQs), which are first-order formulas constructed from atoms $A(x)$, $r(x, y)$, and $x = y$. We use $\text{ans}(\mathcal{I}, q)$ to denote the set of all answers to the FOQ q in the interpretation \mathcal{I} .

A *signature* is a set of concept and role names, which are uniformly called *symbols* in this context. A Σ -*ABox* is an ABox that uses only concept and role names from Σ . The *full signature* is the signature that contains all concept and role names.

Definition 1 (FO-rewritability). *Let \mathcal{T} be an \mathcal{EL} -TBox and Σ an ABox signature. An IQ q is FO-rewritable relative to \mathcal{T} and Σ if there is a FOQ φ such that $\text{cert}_{\mathcal{T}}(\mathcal{A}, q) = \text{ans}(\mathcal{I}_{\mathcal{A}}, \varphi)$ for all Σ -ABoxes \mathcal{A} . Then φ is an FO-rewriting of q relative to \mathcal{T} and Σ .*

Example 1. Recall from the introduction that $A(x)$ is not FO-rewritable relative to $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$ and the full signature. If we add $\exists r.\top \sqsubseteq A$ to \mathcal{T} , then $A(x)$ is FO-rewritable relative to the resulting TBox and the full signature, and $\varphi(x) = A(x) \vee \exists y r(x, y)$ is an FO-rewriting. If we choose $\Sigma = \{A\}$, then $A(x)$ becomes FO-rewritable also relative to the original \mathcal{T} , with the trivial FO-rewriting $A(x)$. Conversely, if a query q is FO-rewritable relative to a TBox \mathcal{T}' and a signature Σ , then q is FO-rewritable relative to \mathcal{T}' and any $\Sigma' \subseteq \Sigma$ (take an FO-rewriting relative to \mathcal{T}' and Σ and replace all atoms which involve predicates that are not in Σ' with false).

Sometimes, instance queries have the more general form $C(x)$ with C an \mathcal{EL} -concept. Since $C(x)$ is FO-rewritable relative to \mathcal{T} and Σ whenever $A(x)$ is FO-rewritable relative to $\mathcal{T} \cup \{A \equiv C\}$ and Σ , A a fresh concept name, queries of this form are captured by the results in this paper.

3 General TBoxes – Upper Bounds

We first characterize failure of FO-rewritability of an IQ $A(x)$ relative to a TBox \mathcal{T} and an ABox signature Σ in terms of the existence of certain Σ -ABoxes and then show how to decide the latter. The following result provides the starting point. An ABox is called *tree-shaped* if the directed graph $(\text{Ind}(\mathcal{A}), \{(a, b) \mid r(a, b) \in \mathcal{A}\})$ is a tree and $r(a, b), s(a, b) \in \mathcal{A}$ implies $r = s$. A FOQ is a *tree-UCQ* if it is a disjunction

$q_1 \vee \dots \vee q_n$ and each q_i is a conjunctive query (CQ) that is tree-shaped (defined in analogy with tree-shaped ABoxes) and where the root is the only answer variable; see e.g. [15] for details on CQs.

Theorem 1. *Let \mathcal{T} be an \mathcal{EL} -TBox, Σ an ABox signature, and $A(x)$ an IQ. Then*

1. *If $A(x)$ is FO-rewritable relative to \mathcal{T} and Σ , then there is a tree-UCQ that is an FO-rewriting of $A(x)$ relative to \mathcal{T} and Σ ;*
2. *If $\varphi(x)$ is an FO-rewriting of $A(x)$ relative to \mathcal{T} and tree-shaped Σ -ABoxes and $\varphi(x)$ is a tree-UCQ, then $\varphi(x)$ is an FO-rewriting relative to \mathcal{T} and Σ ;*
3. *$A(x)$ is FO-rewritable relative to \mathcal{T} and Σ iff $A(x)$ is FO-rewritable relative to \mathcal{T} and tree-shaped Σ -ABoxes.*

Of the three points in Theorem 1, Point 1 is most laborious to prove. It involves applying an Ehrenfeucht-Fraïssé game and explicitly constructing a tree-UCQ as a disjunction of certain \mathcal{EL} -concepts (c.f. the characterization of FO-rewritability in terms of datalog boundedness given in [15] and its proof). Point 2 can then be derived from Point 1, and Point 3 is an immediate consequence of Points 1 and 2.

For a tree-shaped ABox \mathcal{A} and $k \geq 0$, we use $\mathcal{A}|_k$ to denote the restriction of \mathcal{A} to depth k . The following provides the first version of the announced characterization of FO-rewritability in terms of the existence of certain ABoxes.

Theorem 2. *Let \mathcal{T} be an \mathcal{EL} -TBox, Σ an ABox signature, and $A(x)$ an IQ. Then $A(x)$ is not FO-rewritable relative to \mathcal{T} and Σ iff for every $k \geq 0$, there is a tree-shaped Σ -ABox \mathcal{A} of depth exceeding k with root a_0 s.t. $\mathcal{T}, \mathcal{A} \models A(a_0)$ and $\mathcal{T}, \mathcal{A}|_k \not\models A(a_0)$.*

The proof of Theorem 2 builds on Point 1 of Theorem 1. Note that if $A(x)$ is FO-rewritable relative to \mathcal{T} and Σ , then there is a $k \geq 0$ such that for all tree-shaped Σ -ABoxes \mathcal{A} of depth exceeding k with root a_0 , $\mathcal{T}, \mathcal{A} \models A(a_0)$ implies $\mathcal{T}, \mathcal{A}|_k \models A(a_0)$. In the proof of Theorem 2, we explicitly construct FO-rewritings which are tree-UCQs of outdegree at most $|\mathcal{T}|$ and depth at most k .

To proceed, it is convenient to work with TBoxes in *normal form*, where all CIs must be of one of the forms $A \sqsubseteq B_1$, $A \sqsubseteq \exists r.B$, $\top \sqsubseteq A$, $B_1 \sqcap B_2 \sqsubseteq A$, $\exists r.B \sqsubseteq A$ with A, B, B_1, B_2 concept names. This can be assumed without loss of generality:

Lemma 1. *For any \mathcal{EL} -TBox \mathcal{T} , ABox signature Σ , and IQ $A(x)$, there is a TBox \mathcal{T}' in normal form such that for any FOQ φ , we have that $\varphi(x)$ is an FO-rewriting of $A(x)$ relative to \mathcal{T} and Σ iff $\varphi(x)$ is an FO-rewriting of $A(x)$ relative to \mathcal{T}' and Σ .*

To exploit Theorem 2 for building a decision procedure for FO-rewritability, we impose a bound on k . The next theorem is proved using Theorem 2 and a pumping argument.

Theorem 3. *Let \mathcal{T} be an \mathcal{EL} -TBox in normal form, Σ an ABox signature, $A(x)$ an IQ, and $n = |(\text{sig}(\mathcal{T}) \cup \Sigma) \cap \mathbf{N}_C|$. Then $A(x)$ is not FO-rewritable relative to \mathcal{T} and Σ iff there exists a tree-shaped Σ -ABox \mathcal{A} of depth exceeding 2^n with root a_0 such that $\mathcal{T}, \mathcal{A} \models A(a_0)$ and $\mathcal{T}, \mathcal{A}|_{2^n} \not\models A(a_0)$.*

Note that, with the remark after Theorem 2, we obtain a triple exponential upper bound on the size of FO-rewritings. The bound in Theorem 3 is optimal in the sense that, for every $n \geq 1$, there is an \mathcal{EL} -TBox \mathcal{T} and an IQ $A(x)$ such that $|\text{sig}(\mathcal{T}) \cap \mathbf{N}_C| = n$,

$A(x)$ is FO-rewritable relative to \mathcal{T} and the full Σ , and for all ABoxes of depth at least 2^n with root a_0 , we have $\mathcal{T}, \mathcal{A} \models A(a_0)$ iff $\mathcal{T}, \mathcal{A}|_{2^n-1} \models A(a_0)$. Such a \mathcal{T} can be constructed by simulating a binary counter, see Section 4 of [13]. Based on Theorem 3, we can establish the following result.

Theorem 4. *Deciding FO-rewritability of an IQ relative to an \mathcal{EL} -TBox and an ABox signature is in EXPTIME.*

The proof utilizes non-deterministic bottom-up automata on finite, ranked trees: we construct exponential-size automata that accept precisely the ABoxes \mathcal{A} from Theorem 3 and then decide their emptiness in PTIME.

When Σ is full, the characterization given in Theorem 3 can be further improved. An ABox \mathcal{A} is *linear* if it consists of role assertions $r_0(a_0, a_1), \dots, r_{n-1}(a_{n-1}, a_n)$ and concept assertions $A(a)$ with $a \in \{a_0, \dots, a_n\}$. Somewhat unexpectedly, with full Σ we can replace the tree-shaped ABoxes from Theorem 3 with linear ones.

Theorem 5. *Let \mathcal{T} be an \mathcal{EL} -TBox in normal form, $A(x)$ an IQ, and $n = |(\text{sig}(\mathcal{T}) \cup \Sigma) \cap \mathbb{N}_{\mathbb{C}}|$. Then $A(x)$ is not FO-rewritable relative to \mathcal{T} (and the full Σ) iff there exists a linear ABox \mathcal{A} of depth exceeding 2^n with root a_0 such that $\mathcal{T}, \mathcal{A} \models A(a_0)$ and $\mathcal{T}, \mathcal{A}|_{2^n} \not\models A(a_0)$.*

The surprisingly subtle proof of Theorem 5 is based on the careful extraction of a linear ABox from the tree-shaped one whose existence is guaranteed by Theorem 3. The subtlety is largely due to the fact that it is not sufficient to simply select a linear chain of individuals from the tree-shaped ABox; additionally, the concept assertions on that chain have to be modified in a very careful way.

The following example shows that, when Σ is not full, tree-shaped ABoxes in Theorem 3 cannot be replaced with linear ones.

Example 2. Let $\mathcal{T} = \{A_i \sqsubseteq X_i, B_i \sqcap X_i \sqsubseteq Y_i, \exists r.Y_i \sqsubseteq X_i \mid i \in \{1, 2\}\} \cup \{X_1 \sqcap X_2 \sqsubseteq X, B_1 \sqcap B_2 \sqsubseteq Z, \exists r.Z \sqsubseteq X\}$,
 $\Sigma = \{A_1, A_2, B_1, B_2, r\}$, and take the IQ $X(x)$. The tree-shaped ABox

$$\mathcal{A} = \{r(a_0, a_{i,0}), r(a_{i,0}, a_{i,1}), \dots, r(a_{i,2^n-1}, a_{i,2^n}) \mid i \in \{1, 2\}\} \cup \{B(a_{i,0}), \dots, B(a_{i,2^n}), A_i(a_{i,2^n}) \mid i \in \{1, 2\}\},$$

with n as in Theorems 3 and 5, is of depth exceeding 2^n and it can be verified that $\mathcal{T}, \mathcal{A} \models X(a_0)$, but $\mathcal{T}, \mathcal{A}|_{2^n} \not\models X(a_0)$. However, for all linear Σ -ABoxes \mathcal{A} , we have $\mathcal{T}, \mathcal{A} \models X(a_0)$ iff $\mathcal{T}, \mathcal{A}|_1 \models X(a_0)$.

Theorem 5 allows us to replace the non-deterministic tree automata in the proof of Theorem 6 with word automata, improving the upper bound to PSPACE.

Theorem 6. *Deciding FO-rewritability of an IQ relative to an \mathcal{EL} -TBox and the full ABox signature is in PSPACE.*

4 General TBoxes – Lower Bounds

We establish lower bounds that match the upper bounds from the previous section.

Theorem 7. *Deciding FO-rewritability of an IQ relative to a general \mathcal{EL} -TBox and an ABox signature Σ is (1) PSPACE-hard when Σ is full and (2) EXPTIME-hard when Σ is an input.*

The proof of Point 1 is by reduction of the word problem of polynomially space-bounded deterministic Turing machines (DTMs). For Point 2, we modify the proof of Point 1 to yield a reduction of the word problem of polynomially space-bounded alternating Turing machines (ATMs). We start with the former.

Let $M = (Q, \Omega, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ be a DTM and $p(\cdot)$ its polynomial space bound. We assume w.l.o.g. that M terminates on every input, that it never attempts to move left on the left-most end of the tape, that $q_0 \notin \{q_{\text{acc}}, q_{\text{rej}}\}$, and that there are no transitions defined for q_{acc} and q_{rej} . Let $x \in \Omega^*$ be an input to M of length n . Our aim is to construct a TBox \mathcal{T} and select a concept name B such that B is *not* FO-rewritable relative to \mathcal{T} and the full signature iff M accepts x .

By Theorem 5, non-FO-rewritability of B w.r.t. \mathcal{T} is witnessed by a sequence of linear ABoxes of increasing depth. In the reduction, these ABoxes take the form of longer and longer r -chains (with r a role name). The chains represent the computation of M on x , repeated over and over again. Specifically, the tape contents, the current state, and the head position are represented using the elements of $\Gamma \cup (\Gamma \times Q)$ as concept names. If, for example, $x = ab$ and the computation of M on x consists of the two configurations qab and $aq'b$,⁴ then this is represented by ABoxes of the form

$$\{r(b_0, b_1), r(b_1, b_2), r(b_2, b_3), \dots, r(b_{n-1}, b_n)\}$$

where additionally, the concept (q, a) is asserted for b_0, b_4, b_8, \dots , a is asserted for b_1, b_5, b_9, \dots and for b_2, b_6, b_{10}, \dots , and (q', b) for b_3, b_7, b_{11}, \dots . If M accepts x , then B is propagated backwards along these chains (from b_0 to b_1 to b_2 etc) unboundedly far, starting from a single explicit occurrence of B asserted for b_0 . If M rejects x or the chain in the ABox does not properly represent the computation of M on x , then B will already be implied by any subchain of length at most $p(n)^2$ and thus the unbounded propagation of B gets ‘disrupted’ resulting in FO-rewritability of B relative to \mathcal{T} .

The following CI in \mathcal{T} results in backwards propagation of B provided that every ABox individual is labeled with at least one symbol from $\Gamma \cup \Gamma \times Q$:

$$\exists r.(A \sqcap B) \sqsubseteq B \text{ for all } A \in \Gamma \cup (\Gamma \times Q). \quad (1)$$

Disrupt the propagation of B when M does not accept x :

$$(a, q_{\text{rej}}) \sqsubseteq B \text{ for all } a \in \Gamma.$$

We have to enforce that the ABox actually represents a (repeated) computation of M . To do this, we again use disruptions of the propagation of B : whenever an ABox \mathcal{A}

⁴ $uqv \in \Gamma^*Q\Gamma^*$ means that M is in state q , the tape left of the head is labeled with u , and starting from the head position, the remaining tape is labeled with v .

represents a configuration sequence that is not a proper computation, then B is implied by a sub-chain of bounded length. Let forbid denote the set of all tuples (A_1, A_2, A_3, A) with $A_i \in \Gamma \cup (\Gamma \times Q)$ such that whenever three consecutive tape cells in a configuration c are labeled with A_1, A_2, A_3 , then in the successor configuration c' of c , the tape cell corresponding to the middle cell *cannot* be labeled with A . Put

$$A \sqcap \exists r^{p(n)+1}.A_1 \sqcap \exists r^{p(n)}.A_2 \sqcap \exists r^{p(n)-1}.A_3 \sqsubseteq B \quad (2)$$

for all $(A_1, A_2, A_3, A) \in \text{forbid}$. This ensures that the transition relation is respected and that the content of tape cells which are not under the head does not change. We also need to say that every tape cell has a unique label, that there is at not more than one head position per configuration, and at least one, again via disrupting the propagation of B :

$$\begin{aligned} A \sqcap A' &\sqsubseteq B \quad \text{for all distinct } A, A' \in \Gamma \cup (\Gamma \times Q) \\ (a, q) &\sqsubseteq H \quad \text{for all } (a, q) \in \Gamma \times Q & a &\sqsubseteq \overline{H} \quad \text{for all } a \in \Gamma \\ \exists r^i.H \sqcap \exists r^j.H &\sqsubseteq B \quad \text{for } i < j < p(n) & \overline{H} \sqcap \exists r.\overline{H} \sqcap \dots \sqcap \exists r^{p(n)-1}.\overline{H} &\sqsubseteq B \end{aligned}$$

where H is a concept name indicating that the head is on the current cell and \overline{H} indicating that this is not the case. It remains to set up the initial configuration. It is tempting to introduce a concept name I that sets up the initial configuration and must be used at the end of the r -chain to start the propagation of B . However, since we assume the ABox signature Σ to be full, we can always put B itself at the end of the chain, avoiding I . To fix this issue, we refrain from introducing I , but utilize the final states q_{acc} and q_{rej} , enforcing that they must always be followed by the initial configuration. Let $A_1^{(0)}, \dots, A_m^{(0)}$ be the concept names that describe the initial configuration, i.e., when the input x is $x_1 \cdots x_n$, then $A_0^{(0)} = (x_1, q_0)$, $A_i^{(0)} = x_i$ for $2 \leq i \leq n$, and $A_i^{(0)} = x_i$ is the blank symbol for $n < i \leq p(n)$. Now put

$$\exists r^i.q_{\text{acc}} \sqsubseteq A_i^{(0)} \quad \text{and} \quad \exists r^i.q_{\text{rej}} \sqsubseteq A_i^{(0)} \quad \text{for } 1 \leq i \leq p. \quad (3)$$

Note that all witness ABoxes for non-FO-rewritability of B must eventually contain q_{acc} or q_{rej} , thus the initial configuration will be properly set up at some point: ABoxes \mathcal{A} that do not contain these states do not represent a proper computation of M because M reaches q_{acc} or q_{rej} after at most $p(n)$ states, thus the propagation of B is disrupted in \mathcal{A} .

We now come to Point 2 of Theorem 7. When the ABox signature Σ is not required to be full, it is much simpler to set up the initial configuration. Indeed, we can then simply introduce the mentioned concept name I , add $I \sqsubseteq B$ to \mathcal{T} , and set $\Sigma = \Gamma \cup (\Gamma \times Q) \cup \{r, I\}$ to ensure that we must use I to start the propagation of B . We can further replace the CIs (3) above with $\exists r^i.I \sqsubseteq A_i^{(0)}$ and $\exists r^i.I \sqsubseteq A_i^{(0)}$ for $1 \leq i \leq p$ to ensure that I sets up the initial configuration as intended. With this modification, we can adapt the reduction from DTMs to ATMs in a straightforward way, thus improving the PSPACE lower bound to an EXPTIME one. We only give a brief sketch. Assume w.l.o.g. that each configuration of the ATM M has at most two successor configurations. We introduce a second role name $s \in \Sigma$, reserving r for the first successor

of a configuration and s for the second successor. Then the CIs (1) are replaced with

$$\exists r.(A \sqcap B) \sqcap \exists s.(A' \sqcap B) \sqsubseteq B \text{ for all } A, A' \in \Gamma \cup (\Gamma \times Q).$$

resulting in witness ABoxes to take the form of a tree-shaped ATM computation rather than a linear DTM computation. It remains to adapt the CIs (2) to reflect the branching of computations. The set forbid now contains tuples $(A_1, A_2, A_3, A'_1, A'_2, A'_3, A)$, where A_1, A_2, A_3 describe three neighboring cells in the left predecessor configuration and A'_1, A'_2, A'_3 the corresponding cells in the right predecessor configuration (for existential states of M , we simply assume that the left and right predecessor are identical). We can then replace (2) with

$$A \sqcap \exists r^{p(n)+1}. A_1 \sqcap \exists r^{p(n)}. A_2 \sqcap \exists r^{p(n)-1}. A_3 \sqcap \exists s^{p(n)+1}. A'_1 \sqcap \exists s^{p(n)}. A'_2 \sqcap \exists s^{p(n)-1}. A'_3 \sqsubseteq B.$$

5 Classical TBoxes

A classical TBox \mathcal{T} is a finite set of *concept definitions* $A \equiv C$ and CIs $A \sqsubseteq C$ where A is a concept name. No concept name is allowed to occur more than once on the left hand side of a statement in \mathcal{T} .

Theorem 8. *Deciding FO-rewritability of an IQ relative to a classical \mathcal{EL} -TBox and an ABox signature is in PTIME.*

We give examples illustrating FO-rewritability in classical TBoxes and give the main idea of the proof.

Example 3. (a) The IQ $A(x)$ is not FO-rewritable relative to the classical TBox $\{A \equiv \exists r.A\}$ and the full ABox signature.

(b) The concept name A has a cyclic definition in the TBox $\mathcal{T} = \{A \equiv B \sqcap \exists r.A, B \sqsubseteq \exists r.A\}$ which often indicates non-FO-rewritability, but in this case the IQ $A(x)$ has an FO-rewriting relative to \mathcal{T} and the full ABox signature, namely $\varphi(x) = A(x) \vee B(x)$.

To present the idea of the proof, we use an appropriate normal form for classical TBoxes. A concept name A is *defined in* \mathcal{T} if there is a definition $A \equiv C \in \mathcal{T}$ and *primitive* otherwise; A is *non-conjunctive* in \mathcal{T} if it occurs in \mathcal{T} , but there is no CI of the form $A \equiv B_1 \sqcap \dots \sqcap B_n$ in \mathcal{T} with $n \geq 1$ and B_1, \dots, B_n concept names. We use $\text{non-conj}(\mathcal{T})$ to denote the set of non-conjunctive concepts in \mathcal{T} . A classical TBox \mathcal{T} is in *normal form* if it is a set of statements $A \equiv \exists r.B$ and $A \equiv B_1 \sqcap \dots \sqcap B_n$ where B, B_1, \dots, B_n are concept names and B_1, \dots, B_n are non-conjunctive. For every classical TBox \mathcal{T} , one can construct in polynomial time a classical TBox \mathcal{T}' in normal form that uses additional concept names such that $\mathcal{T}' \models \mathcal{T}$ and every model of \mathcal{T} can be expanded to a model of \mathcal{T}' [10]. It is not hard to verify that an IQ $A(x)$ is FO-rewritable relative to \mathcal{T} and Σ if and only if it is FO-rewritable relative to \mathcal{T}' and Σ , provided that A is not among the new concept names introduced during the construction of \mathcal{T}' . For a classical TBox \mathcal{T} in normal form and a concept name A , define

$$\text{non-conj}_{\mathcal{T}}(A) = \begin{cases} \{A\} & \text{if } A \text{ is non-conjunctive in } \mathcal{T} \\ \{B_1, \dots, B_n\} & \text{if } A \equiv B_1 \sqcap \dots \sqcap B_n \in \mathcal{T}. \end{cases}$$

Our polytime algorithm utilizes an ABox introduced in [10, 9] in the context of conservative extensions and logical difference: given a classical TBox \mathcal{T} in normal form and an ABox signature Σ , we compute in polytime a polysize Σ -ABox $\mathcal{A}_{\mathcal{T},\Sigma}$ with individual names a_B , B non-conjunctive in \mathcal{T} , such that for any Σ -ABox \mathcal{A} , individual name a in \mathcal{A} , and concept name A the following conditions are equivalent:

- $\mathcal{T}, \mathcal{A} \not\models A(a)$;
- there exists $B \in \text{non-conj}_{\mathcal{T}}(A)$ such that (\mathcal{A}, a) is simulated by $(\mathcal{A}_{\mathcal{T},\Sigma}, a_B)$ (see appendix of long version of this paper).

It follows that to check whether $A(x)$ is FO-rewritable, instead of considering arbitrary tree-shaped Σ -ABoxes \mathcal{A} and $\mathcal{A}|_k$ as in Theorem 2, it suffices to consider the tree unfolding of $\mathcal{A}_{\mathcal{T},\Sigma}$ at a_B and its restriction to depth k , for all $B \in \text{non-conj}_{\mathcal{T}}(A)$. The original search problem has been reduced to the problem of analysing the tree unfolding of $\mathcal{A}_{\mathcal{T},\Sigma}$. A polytime algorithm performing that analysis is given in the long version.

6 Semi-Acyclic TBoxes

It is easy to see that every IQ is FO-rewritable relative to every acyclic \mathcal{EL} -TBox and every ABox signature Σ . We observe that the same holds for semi-acyclic TBoxes, where some cycles are still allowed, and that it is possible to find rewritings of polynomial size when only databases of domain size at least two are admitted.

A *semi-acyclic TBox* is defined like a classical TBox, except that definitorial cycles are disallowed, i.e., there cannot be concept definitions $A_0 \equiv C_0, \dots, A_{n-1} \equiv C_{n-1}$ such that A_i occurs in $C_{i+1 \bmod n}$. Note that cycles via concept *inclusions*, such as $A \sqsubseteq \exists r.A$, are still permitted. Let \mathcal{T} be a semi-acyclic TBox and Σ an ABox signature. For an \mathcal{EL} -concept C , we use $\text{pre}_{\mathcal{T},\Sigma}(C)$ to denote the FO-formula $\bigvee_{B \in \Sigma \mid \mathcal{T} \models B \sqsubseteq C} B(x)$. For all concept names A and \mathcal{EL} -concepts C and D and role names r , set

$$\begin{aligned}
\varphi_{\top, \mathcal{T}}^{\Sigma}(x) &= \text{true} \\
\varphi_{A, \mathcal{T}}^{\Sigma}(x) &= \text{pre}_{\mathcal{T}, \Sigma}(A) && \text{if } A \text{ is primitive} \\
\varphi_{A, \mathcal{T}}^{\Sigma}(x) &= \varphi_{C, \mathcal{T}}^{\Sigma}(x) && \text{if } A \equiv C \in \mathcal{T} \\
\varphi_{C \sqcap D, \mathcal{T}}^{\Sigma}(x) &= \varphi_{C, \mathcal{T}}^{\Sigma}(x) \wedge \varphi_{D, \mathcal{T}}^{\Sigma}(x) \\
\varphi_{\exists r.C, \mathcal{T}}^{\Sigma}(x) &= \text{pre}_{\mathcal{T}, \Sigma}(\exists r.C) \vee \exists y.(r(x, y) \wedge \varphi_{C, \mathcal{T}}^{\Sigma}[y/x]) && \text{if } r \in \Sigma \\
\varphi_{\exists r.C, \mathcal{T}}^{\Sigma}(x) &= \text{pre}_{\mathcal{T}, \Sigma}(\exists r.C) && \text{if } r \notin \Sigma
\end{aligned}$$

where $\varphi[x/y]$ denotes the result of first renaming all bound variables in φ so that y does not occur, and then replacing the free variable x of φ with y .

Lemma 2. *For all IQs $A(x)$, $\varphi_{A, \mathcal{T}}^{\Sigma}(x)$ is an FO-rewriting of $A(x)$ relative to \mathcal{T} and Σ .*

The size of $\varphi_{A, \mathcal{T}}^{\Sigma}(x)$ can clearly be exponential in the size of \mathcal{T} , for example when $A = A_n$ and $\mathcal{T} = \{A_i \equiv \exists r.A_{i-1} \sqcap \exists s.A_{i-1} \mid 1 \leq i \leq n\}$. To reduce $\varphi_{A, \mathcal{T}}^{\Sigma}$ to polynomial size, we can use Avigad's observation that FO supports structure sharing [1]. More precisely, let φ be a positive FOQ (such as $\varphi_{A, \mathcal{T}}^{\Sigma}$) whose subformulas

include $\psi(x_1), \dots, \psi(x_n)$. The multiple occurrences of ψ can be avoided by rewriting φ to $\exists u \forall y \forall z ((\psi(y) \leftrightarrow z = u) \rightarrow \varphi')$ where φ' is φ with each $\psi(x_i)$ replaced with $y = x_i \rightarrow z = u$. Intuitively, we iterate over all y and memorize whether $\psi(y)$ holds using identity of z with u . Since we need at least two different ‘values’ for z to make this trick work, the resulting FOQ is an FO-rewriting only on ABoxes with at least two individual names.

7 Related Work

In [15], deciding FO-rewritability is studied in the context of the expressive DL \mathcal{ALCFI} and several of its fragments. In general, though, the setup in that paper is different: while we are interested in deciding FO-rewritability of a single query relative to a TBox, the results in [15] concern deciding whether, for a given TBox \mathcal{T} , *all* queries are FO-rewritable relative to \mathcal{T} . It is shown that this problem is decidable for Horn- \mathcal{ALCFI} -TBoxes of depth at most two and for Horn- \mathcal{ALCF} -TBoxes (queries are IQs or, equivalently, CQs). As a by-product of these results, a close connection between FO-rewritability of TBoxes formulated in Horn DLs and boundedness of datalog programs is observed, see e.g. [6, 17] for the latter problem. In its original formulation, the following result is established for a larger class of TBoxes, namely materializable \mathcal{ALCFI} -TBoxes of depth one.

Lemma 3 ([15]). *For every (general) \mathcal{EL} -TBox \mathcal{T} in normal form, there is a datalog program $\Pi_{\mathcal{T}}$ such that for every ABox signature Σ and IQ $A(x)$, the predicate A is bounded in $\Pi_{\mathcal{T}}$ relative to Σ -databases iff $A(x)$ is FO-rewritable relative to \mathcal{T} and Σ .*

In [15], the program $\Pi_{\mathcal{T}}$ is of exponential size. Since we are only interested in \mathcal{EL} -TBoxes, it is easy to find a $\Pi_{\mathcal{T}}$ of polynomial size. More specifically, $\Pi_{\mathcal{T}}$ consists of

$$\begin{aligned} A(x) &\leftarrow \text{true} && \text{if } \top \sqsubseteq A \in \mathcal{T} \\ B(x) &\leftarrow r(x, y), A(x) && \text{if } \exists r.A \sqsubseteq B \in \mathcal{T} \quad X_{\exists r.A}(x) \leftarrow B(x) \text{ if } B \sqsubseteq \exists r.A \in \mathcal{T} \\ B(x) &\leftarrow A_1(x), A_2(x) && \text{if } A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T} \quad (\text{where possibly } A_1 = A_2) \\ B(x) &\leftarrow X_{\exists r.A}(x) && \text{if } \exists r.B_0 \sqsubseteq B \in \mathcal{T} \text{ and } \mathcal{T} \models A \sqsubseteq B_0 \end{aligned}$$

This allows to carry over the 2EXPTIME upper bound for predicate boundedness of connected monadic datalog programs [6] to FO-rewritability of an IQ relative to a general \mathcal{EL} -TBoxes and an ABox signature.⁵

Note that boundedness has been studied also in the context of the μ -calculus and monadic second order (MSO) logic [16, 4]. Here, an EXPTIME upper bound is known from [16] and it seems likely that this result can be utilized to find an alternative proof of Theorem 6. In particular, it is possible to find a μ -calculus rewriting φ of an IQ $A(x)$ relative to an \mathcal{EL} -TBox \mathcal{T} and ABox signature Σ : proceeding similarly to the construction of the above datalog program $\Pi_{\mathcal{T}}$, we can find a μ -calculus formula $\varphi_{\mathcal{T}, \Sigma}$ such that for all Σ -ABoxes \mathcal{A} and $a \in \text{Ind}(\mathcal{A})$, we have $\mathcal{T}, \mathcal{A} \models A(a)$ iff $\mathcal{I}_{\mathcal{A}}, a \models \varphi$. When simultaneous fixpoints are admitted, φ even has polynomial size.

⁵ Note that we explicitly fix the signature Σ of the databases over which boundedness of $\Pi_{\mathcal{T}}$ is considered instead of assuming that only the EDB predicates can be used in the data as in [6]; this is only for simplicity and, in fact, it is easy to adapt $\Pi_{\mathcal{T}}$ to the latter assumption.

8 Conclusions

It would be interesting to generalize the results presented in this paper to more expressive DLs and to more expressive query languages. Regarding the former, we note that using the techniques in [14, 15] it is possible to derive a NEXPTIME upper bound for deciding FO-rewritability of IQs relative to Horn- \mathcal{ALCI} -TBoxes and ABox signatures. Regarding the latter, CQs are a natural choice and we believe that a mix of techniques from this paper and those in [3] might provide a good starting point. It is interesting to note that FO-rewritability of all IQ-atoms $A(x)$ in a CQ q does not imply that q is FO-rewritable and the converse fails, too.

Acknowledgements. C. Lutz was supported by the DFG SFB/TR 8 ‘Spatial Cognition’.

References

1. J. Avigad. Eliminating definitions and skolem functions in first-order logic. In Proc. of LICS, pages 139–146. IEEE Computer Society, 2001.
2. F. Baader, M. Bienvenu, C. Lutz, and F. Wolter. Query and predicate emptiness in description logics. In Proc. of KR. AAAI Press, 2010.
3. M. Bienvenu, C. Lutz, and F. Wolter. Query containment in description logics reconsidered. In Proc. of KR, 2012. To appear.
4. A. Blumensath, M. Otto, and M. Weyer. Decidability results for the boundedness problem. Manuscript, 2012.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning, 39(3):385–429, 2007.
6. S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi. Decidable optimization problems for database logic programs. In Proc. of STOC, pages 477–490. ACM, 1988.
7. G. Gottlob and T. Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In Proc. of DL. CEUR-WS, 2011.
8. S. Kikot, R. Kontchakov, V. V. Podolskii, and M. Zakharyashev. Exponential lower bounds and separation for query rewriting. CoRR, abs/1202.4193, 2012.
9. B. Konev, M. Ludwig, D. Walther, and F. Wolter. The logical diff for the lightweight description logic \mathcal{EL} . Technical report, U. of Liverpool, <http://www.liv.ac.uk/~frank/publ/>, 2011.
10. B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. In Proc. of IJCAR, pages 259–274. Springer, 2008.
11. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyashev. The combined approach to query answering in DL-Lite. In Proc. of KR. AAAI Press, 2010.
12. C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic \mathcal{EL} using a relational database system. In Proc. of IJCAI, pages 2070–2075. AAAI Press, 2009.
13. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic \mathcal{EL} . In J. of Symbolic Computation 45(2): 194–228, 2010.
14. C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In Proc. of DL. CEUR-WS, 2011.
15. C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In Proc. of KR, 2012. To appear.
16. M. Otto. Eliminating recursion in the μ -calculus. In Proc. of STACS, pages 531–540. Springer, 1999.
17. R. van der Meyden. Predicate boundedness of linear monadic datalog is in PSPACE. Int. J. Found. Comput. Sci., 11(4):591–612, 2000.

Answering Expressive Path Queries over Lightweight DL Knowledge Bases ^{*}

Meghyn Bienvenu¹, Magdalena Ortiz², and Mantas Šimkus²

¹ LRI - CNRS & Université Paris Sud

² Institute of Information Systems, Vienna University of Technology

Abstract. We establish tight complexity bounds for answering an extension of conjunctive 2-way regular path queries over \mathcal{EL} and DL-Lite knowledge bases.

1 Introduction

It has been extensively argued in the description logic (DL) community that answering queries over ABoxes in the presence of ontological constraints formulated in a DL TBox is a fundamental reasoning service. In databases, similar attention has been paid to the related problem of querying graph databases, which are relational databases where only unary and binary predicates occur, or in other words, node- and edge-labeled graphs [8, 3]. The relevance of both problems lies in the fact that in many application areas data can be naturally modeled as an ABox or a graph database. This applies, in particular, to XML data on the web, including RDF datasets. While the two communities share some common research goals, the research agendas they have pursued differ significantly. In the DL community, the focus has been on designing efficient algorithms for answering (plain) conjunctive queries in the presence of expressive ontological constraints. By contrast, work on graph databases typically does not consider ontological knowledge, but instead aims at supporting expressive query languages, like regular path queries (RPQs) and their extensions, which enable sophisticated navigation of paths.

This paper aims to help bridge this gap, by considering an expressive extension of RPQs, and providing algorithms and precise complexity bounds for the \mathcal{EL} and DL-Lite families of lightweight DLs. We build on *conjunctive (2-way) regular path queries* (C2RPQs), which simultaneously extend plain conjunctive queries (CQs) and basic RPQs: they allow conjunctions of atoms that can share variables in arbitrary ways, where the atoms may contain regular expressions that navigate the arcs of the database (roles) in both directions. C2RPQs are one of the most expressive and popular languages for querying graph databases. These queries have already been studied for some DLs. In particular, automata-based algorithms have been proposed for the very expressive DLs \mathcal{ZIQ} , \mathcal{ZIO} , and \mathcal{ZOQ} [5, 6], for which query answering is 2-EXPTIME hard. Even in data complexity, that is, when the query and ontology are assumed fixed, these algorithms need exponential time. More recently, algorithms for answering C2RPQs were proposed in [11] for Horn- \mathcal{SHOIQ} and Horn- \mathcal{SRIOQ} . They are polynomial in data complexity but still EXPTIME in combined, which is worst-case optimal for these

^{*} This work partially supported by the Austrian Science Fund (FWF) grants P20840 and T515.

logics. For prominent lightweight DLs like the DL-Lite [4] and \mathcal{EL} [2], which underly the OWL 2 profiles, queries with regular paths had not been explored and many questions remained open, like whether algorithms that require only polynomial space are possible. In DL-Lite, FO-rewritability and AC_0 data complexity are clearly lost, since we can express reachability, but it was not known whether P-hardness was avoidable. In this paper, we answer these questions by providing precise complexity bounds.

We propose an extension of C2RPQs that we call *conjunctive (2-way) regular path queries with complex labels*, abbreviated ℓ -C2RPQs. To illustrate its expressiveness, we consider a graph representation of the *Mathematics Genealogy Project (MGP)* database, which contains about 160K historic records of advisor relationships of PhD holders in mathematics and related disciplines. We use nodes for mathematicians, theses, topics of research, and universities. Nodes and edges are labeled with concepts (unary relations) and roles (binary relations), respectively. Figure 1 depicts a fragment of such a graph.

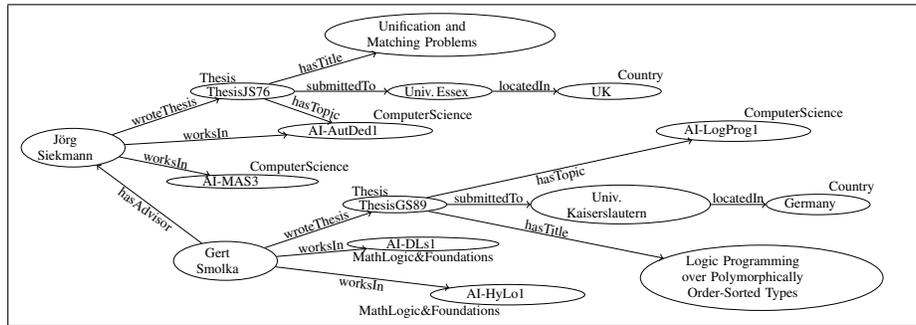


Fig. 1: Example graph database of the Mathematics Genealogy Project

An ontology containing the axioms in Fig. 2 can be used to express, for example, that a person that works in computer science or that wrote a doctoral thesis in computer science is a computer scientist (1,2). In a similar way we can define other specialities such as biologists (3,4), logicians (5,6), physicists, etc. We group the first level subjects of the Mathematics Subject Classification (MSC) used in the MGP database into their 5 major areas (7–11), and these major areas into subjects (12–16).

(1)	$\exists \text{worksOn. CompSci} \sqsubseteq \text{CompScientist}$	
(2)	$\exists \text{wroteThesis.} (\exists \text{hasTopic. CompSci}) \sqsubseteq \text{CompScientist}$	
(3)	$\exists \text{worksOn. Biology\&NaturalSciences} \sqsubseteq \text{Biologist}$	
(4)	$\exists \text{wroteThesis.} (\exists \text{hasTopic. Biology\&NaturalSciences}) \sqsubseteq \text{Biologist}$	
(5)	$\exists \text{worksOn. MathLogic\&Foundns} \sqsubseteq \text{Logician}$	
(6)	$\exists \text{wroteThesis.} (\exists \text{hasTopic. MathLogic\&Foundns}) \sqsubseteq \text{Logician}$	
(7)	$\text{MathLogic\&Foundns} \sqsubseteq \text{General\&Foundns}$	(12) $\text{General\&Foundns} \sqsubseteq \text{Subject}$
(8)	$\text{Geometry} \sqsubseteq \text{Geometry\&Topology}$	(13) $\text{DiscreteMath\&Algebra} \sqsubseteq \text{Subject}$
(9)	$\text{CompSci} \sqsubseteq \text{AppliedMath\&Other}$	(14) $\text{Analysis} \sqsubseteq \text{Subject}$
(10)	$\text{Biology\&NaturalSciences} \sqsubseteq \text{AppliedMath\&Other}$	(15) $\text{Geometry\&Topology} \sqsubseteq \text{Subject}$
(11)	$\text{Physics} \sqsubseteq \text{AppliedMath\&Other}$	(16) $\text{AppliedMath\&Other} \sqsubseteq \text{Subject}$

Fig. 2: An example MGP ontology

In RPQs, C2RPQs and similar languages, regular paths usually talk about the arc labels only, and do not allow to verify conditions on the node labels. For example, one can use the expression $\text{hasAdvisor}^* \circ \text{WroteThesis} \circ \text{hasTopic}$ to navigate arbitrarily long chains of advisors and visit their thesis topics, but we cannot look at the subjects and thesis topics and impose conditions on them. This limitation is not so significant for

graph databases, as arc labels play a more prominent role and are often used to simulate node labels. In the DL setting, by contrast, concepts are crucial and should be treated as first-class citizens. For this reason, we add to C2RPQs the ability to talk about combinations of concept and roles that appear along a path. In our language, we can use the expression $(hasAdvisor, Logician \vee CompScientist)^* \circ WroteThesis \circ (hasTopic, Geometry)$ to navigate a chain of advisors that are computer scientist or logicians, until we reach one that wrote a doctoral thesis in Geometry. Note that this query could be expressed (less succinctly) using the *test* operator (cf. [6]): $(hasAdvisor \circ Logician? \cup hasAdvisor \circ CompScientist?)^* \circ WroteThesis \circ hasTopic \circ Geometry?$. However, our language is slightly more expressive (for DLs without role conjunction), since we can navigate a chain of people that are both advisors and coauthors using $(hasAdvisor \wedge coAuthor)^*$.

In this paper, we show that answering these queries over \mathcal{EL} and DL-Lite knowledge bases is PSPACE-complete, but drops to NP if we consider DL-Lite_{RDFS}. For data complexity, the problem is NLSpace-complete for DL-Lite and P-complete for \mathcal{EL} .

2 Preliminaries

We briefly recall the syntax of DL-Lite_R [4] and \mathcal{ELH} [2] (and relevant sublogics). As usual, we assume sets N_C , N_R , and N_I of concept names, role names, and individuals. We will use $\overline{N_R}$ to refer to $N_R \cup \{r^- \mid r \in N_R\}$, and if $R \in \overline{N_R}$, we use R^- to mean r^- if $R = r$ and r if $R = r^-$. An ABox is a set of assertions of the form $A(b)$ or $r(b, c)$, where $A \in N_C$, $r \in N_R$, and $b, c \in N_I$. A TBox is a set of inclusions, whose form depends on the DL in question. In DL-Lite, inclusions take the form $B_1 \sqsubseteq (\neg)B_2$, where each B_i is either A (where $A \in N_C$) or $\exists R$ (where $R \in \overline{N_R}$). DL-Lite_R additionally allows role inclusions of the form $R_1 \sqsubseteq (\neg)R_2$, where $R_1, R_2 \in \overline{N_R}$. DL-Lite_{RDFS} is obtained from DL-Lite_R by disallowing inclusions which contain negation or have existential concepts ($\exists R$) on the right-hand side. In \mathcal{EL} , inclusions have the form $C_1 \sqsubseteq C_2$, where C_1, C_2 are complex concepts constructed as follows: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$. The DL \mathcal{ELH} additionally allows role inclusions of the form $r \sqsubseteq s$, where $r, s \in N_R$. A knowledge base (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox \mathcal{T} and an ABox \mathcal{A} .

As usual, the semantics is based upon interpretations, which take the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ maps each $a \in N_I$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each $A \in N_C$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each $r \in N_R$ to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is straightforwardly extended to general concepts and roles, e.g. $(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$ and $(\exists r.C)^{\mathcal{I}} = \{c \mid \exists d : (c, d) \in r^{\mathcal{I}}, d \in C^{\mathcal{I}}\}$. \mathcal{I} satisfies $G \sqsubseteq H$ if $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$; it satisfies $A(a)$ (resp. $r(a, b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$). \mathcal{I} is a *model* of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if \mathcal{I} satisfies all inclusions in \mathcal{T} and assertions in \mathcal{A} .

To simplify the presentation, we will assume that \mathcal{ELH} TBoxes are *normalized*, meaning that all concept inclusions are of one of the following forms:

$$\top \sqsubseteq A \quad A \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad B_1 \sqcap B_2 \sqsubseteq A \quad \exists r.B \sqsubseteq A$$

with A, B, B_1, B_2 concept names. It is well-known that for every \mathcal{ELH} TBox \mathcal{T} , one can construct in polynomial time a normalized \mathcal{ELH} TBox \mathcal{T}' that uses fresh concept names such that $\mathcal{T}' \models \mathcal{T}$ and every model of \mathcal{T} can be expanded to a model of \mathcal{T}' .

For convenience, we introduce a set of *basic concepts*, denoted BC, defined as follows: $BC = N_C \cup \{\exists R \mid R \in \overline{N_R}\}$ for DL-Lite_R, and $BC = N_C$ for \mathcal{ELH} .

Canonical Models We recall the definition of canonical models for DL-Lite \mathcal{R} and \mathcal{ELH} KBs. For both logics, the domain of the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ for a KB $(\mathcal{T}, \mathcal{A})$ will consist of *paths* of the form $aR_1C_1 \dots R_nC_n$ ($n \geq 0$), where $a \in \text{Ind}(\mathcal{A})$, each C_i is a basic concept, and each R_i a (possibly inverse) role. When \mathcal{T} is a DL-Lite \mathcal{R} TBox, the domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ contains exactly those paths $aR_1\exists R_1^- \dots R_n\exists R_n^-$ which satisfy:

- if $n \geq 1$, then $\mathcal{T}, \mathcal{A} \models \exists R_1(a)$;
- for $1 \leq i < n$, $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_i^- \neq R_{i+1}$.

When \mathcal{T} is a (normalized) \mathcal{ELH} TBox, the domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ contains exactly those paths $ar_1A_1 \dots r_nA_n$ for which each $r_i \in \mathbb{N}_R$, and:

- if $n \geq 1$, then $\mathcal{T}, \mathcal{A} \models \exists r_1.A_1(a)$;
- for $1 \leq i < n$, $\mathcal{T} \models A_i \sqsubseteq \exists r_{i+1}.A_{i+1}$.

We denote the last concept in a path p by $\text{tail}(p)$, and define $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ by taking:

$$\begin{aligned} a^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= a \text{ for all } a \in \text{Ind}(\mathcal{A}) \\ A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{a \in \text{Ind}(\mathcal{A}) \mid \mathcal{T}, \mathcal{A} \models A(a)\} \cup \{p \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \text{Ind}(\mathcal{A}) \mid \mathcal{T} \models \text{tail}(p) \sqsubseteq A\} \\ r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{(a, b) \mid r(a, b) \in \mathcal{A}\} \cup \\ &\quad \{(p_1, p_2) \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \times \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \mid p_2 = p_1 \cdot SC \text{ and } \mathcal{T} \models S \sqsubseteq r\} \cup \\ &\quad \{(p_2, p_1) \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \times \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \mid p_2 = p_1 \cdot SC \text{ and } \mathcal{T} \models S \sqsubseteq r^-\} \end{aligned}$$

Note that $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is composed of a core consisting of the ABox individuals and an *anonymous part* consisting of (possibly infinite) trees rooted at ABox individuals. We will use $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_e$ to denote the submodel of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ obtained by restricting the universe to paths having e as a prefix.

Regular Languages We assume the reader is familiar with regular languages, represented either by regular expressions or nondeterministic finite state automata (NFAs). An NFA over an *alphabet* Σ is a tuple $\alpha = \langle S, \Sigma, \delta, s_0, F \rangle$, where S is a finite set of *states*, $\delta \subseteq S \times \Sigma \times S$ the *transition relation*, $s_0 \in S$ the *initial state*, and $F \subseteq S$ the set of *final states*. We use $L(\alpha)$ to denote the language defined by an NFA α , and when the way a regular language is represented is not relevant, we denote it simply by L .

3 Conjunctive Regular Path Queries with Complex Labels

We now formally introduce our query language.

Definition 1. By $\mathcal{B}(S)$ we denote the set of all (positive) Boolean formulas built from the symbols in $S \cup \{\mathbf{true}, \mathbf{false}\}$ using the connectives \wedge and \vee . A *conjunctive (two-way) regular path query with complex labels* (abbreviated to ℓ -C2RPQ) has the form $q(\mathbf{x}) = \exists \mathbf{y} \varphi$ where \mathbf{x} and \mathbf{y} are tuples of variables, and φ is a conjunction of atoms of the following forms:

- (i) $\beta(t)$, where $\beta \in \mathcal{B}(\mathbb{N}_C)$ and $t \in \mathbb{N}_1 \cup \mathbf{x} \cup \mathbf{y}$, and
- (ii) $L(t, t')$, where L is (an NFA or regular expression defining) a regular language over $\mathcal{B}(\overline{\mathbb{N}_R}) \times \mathcal{B}(\mathbb{N}_C)$, and $t, t' \in \mathbb{N}_1 \cup \mathbf{x} \cup \mathbf{y}$.

As usual, variables and individuals are called terms, and the variables in \mathbf{x} are called answer variables. A query with no answer variables is called Boolean.

If all atoms of type (i) are of the form $A(t)$ with $A \in \mathbb{N}_C$, and the regular languages L in atoms of type (ii) comprise only symbols of the form (R, \mathbf{true}) with $R \in \overline{\mathbb{N}_R}$, then

q is called a conjunctive (two-way) regular path query (C2RPQ). Conjunctive one-way regular path queries with complex labels (ℓ -CRPQs) and conjunctive one-way regular path queries (CRPQs) are defined analogously, the only difference being that they use formulas from $\mathcal{B}(\mathbb{N}_R)$ instead of $\mathcal{B}(\overline{\mathbb{N}_R})$ in atoms of type (ii). Conjunctive queries (CQs) are C2RPQs where all atoms of type (ii) have the form $(R, \mathbf{true})(t, t')$ with $R \in \overline{\mathbb{N}_R}$.

Example 1. For readability, we write symbols $(R, \mathbf{true}) \in \mathcal{B}(\overline{\mathbb{N}_R}) \times \mathcal{B}(\mathbb{N}_C)$ simply as R . Recall the MGP database. The query $q_1(x, y)$ in Fig. 3 searches for pairs of computer scientists that have a biologist as common academic ancestor, and such that there is a physicist on that path. It returns, among others, Jack Minker and Edmund Clarke, who have the biologist and mathematician Johann Bernoulli (1667–1748) as common ancestor on a path including the physicist and mathematician Joseph Fourier (1768 – 1830); as well as Gert Smolka and Georg Gottlob, who have as ancestors Nikolaus Poda von Neuhaus, an Austrian entomologist (1723 - 1798), and the physicist Ludwig Boltzmann (1844 – 1906). The query $q_2(x, y)$ searches for a common academic ancestor x of Robert Kowalski and Franz Baader, together with the country y where the ancestor’s thesis was defended; it requires all ancestors on the path to be computer scientists and logicians. It returns one tuple: (Bernard Meltzer, UK). The query $q_3(x)$ is similar but we require x to be a biologist or physicist, and additionally allow physicists along the path. This query retrieves 8 people, going back to Gabriel Gruber (1740–1805), a Jesuit priest, philosopher, mathematician and professor of physics.

$$\begin{aligned}
q_1(x, y) &= \text{CompScientist} \vee \text{Logician}(x), \text{CompScientist} \vee \text{Logician}(y), \\
&\quad [\text{hasAdvisor}^* \circ (\text{hasAdvisor}, \text{Physicist}) \circ \text{hasAdvisor}^* \circ (\text{hasAdvisor}, \text{Biologist}) \\
&\quad \circ (\text{hasAdvisor}^-)^* \circ (\text{hasAdvisor}^-, \text{Physicist}) \circ (\text{hasAdvisor}^-)^*](x, y) \\
q_2(x, y) &= [(\text{hasAdvisor}, \text{CompScientist} \wedge \text{Logician})^*](\text{RKowalski}, x), \\
&\quad [(\text{hasAdvisor}, \text{CompScientist} \wedge \text{Logician})^*](\text{FBaader}, x) \\
&\quad [\text{wroteThesis} \circ \text{submittedTo} \circ \text{locatedIn}](x, y) \\
q_3(x) &= [(\text{hasAdvisor}, ((\text{CompScientist} \wedge \text{Logician}) \vee \text{Physicist}))^* \\
&\quad \circ (\text{hasAdvisor}, \text{Biologist} \vee \text{Physicist})](\text{RKowalski}, x) \\
&\quad [(\text{hasAdvisor}, ((\text{CompScientist} \wedge \text{Logician}) \vee \text{Physicist}))^* \circ (\text{hasAdvisor})](\text{FBaader}, x)
\end{aligned}$$

Fig. 3: Example queries

We now define the semantics of ℓ -C2RPQs. We say that a set $\mathcal{X} \subset X$ satisfies a formula $\varphi \in \mathcal{B}(X)$, written $\mathcal{X} \models \varphi$, if the formula that results from replacing each $v \in X$ by \mathbf{true} if $v \in \mathcal{X}$ and by \mathbf{false} otherwise is equivalent to \mathbf{true} . For a regular language L over the alphabet $\mathcal{B}(\overline{\mathbb{N}_R}) \times \mathcal{B}(\mathbb{N}_C)$, we call d_2 an L -successor of d_1 in \mathcal{I} if there is some $w = (\Gamma_1, \Upsilon_1) \dots (\Gamma_n, \Upsilon_n) \in L$ and some sequence e_0, \dots, e_n of elements in $\Delta^{\mathcal{I}}$ such that $e_0 = d_1$, $e_n = d_2$, and, for all $1 \leq i \leq n$:

$$\{R \in \overline{\mathbb{N}_R} \mid \langle e_{i-1}, e_i \rangle \in R^{\mathcal{I}}\} \models \Gamma_i \quad \text{and} \quad \{A \in \mathbb{N}_C \mid e_i \in A^{\mathcal{I}}\} \models \Upsilon_i.$$

A *match* for a Boolean ℓ -C2RPQ q in an interpretation \mathcal{I} is a mapping π from the terms in q to elements in $\Delta^{\mathcal{I}}$ such that:

- $\pi(c) = c^{\mathcal{I}}$ if $c \in \mathbb{N}_I$,
- $\{A \in \mathbb{N}_C \mid \pi(t) \in A^{\mathcal{I}}\} \models \beta$ for each atom $\beta(t)$ in q , and
- $\pi(t')$ is an L -successor of $\pi(t)$ for each atom $L(t, t')$ in q .

We write $\mathcal{I} \models q$ if there is a match for q in \mathcal{I} , and $\mathcal{T}, \mathcal{A} \models q$ if $\mathcal{I} \models q$ for every model \mathcal{I} of \mathcal{T}, \mathcal{A} .

Given an ℓ -C2RPQ q with answer variables v_1, \dots, v_k , we say that a tuple of individuals (a_1, \dots, a_k) is a *certain answer* for q w.r.t. \mathcal{T}, \mathcal{A} just in the case that in every model \mathcal{I} of \mathcal{T}, \mathcal{A} there is a match π for q such that $\pi(v_i) = a_i^{\mathcal{I}}$ for every $1 \leq i \leq k$. Just as for CQs and C2RPQs, deciding whether a tuple of individuals is a certain answer for an ℓ -C2RPQ can be linearly reduced to Boolean ℓ -C2RPQ entailment. For this reason, we consider only the latter problem in what follows.

It is well known that the canonical model $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ can be homomorphically embedded into any model of \mathcal{T}, \mathcal{A} , hence a CQ q is entailed by \mathcal{T}, \mathcal{A} if and only if there is a match for q in $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$. This result can be easily lifted from CQs to ℓ -C2RPQs, as ℓ -C2RPQs are also monotonic and their matches are preserved under homomorphisms.

Lemma 1. *For every DL-Lite $_{\mathcal{R}}$ or \mathcal{ELH} KB $(\mathcal{T}, \mathcal{A})$ and Boolean ℓ -C2RPQ q : $\mathcal{T}, \mathcal{A} \models q$ if and only if $\mathcal{I}_{\mathcal{T}, \mathcal{A}} \models q$.*

This property will be a crucial element in establishing our main theorem:

Theorem 1. *Boolean ℓ -C2RPQ entailment is NLSpace-complete in data complexity and PSpace-complete in combined complexity for DL-Lite $_{\mathcal{R}}$; the combined complexity drops to NP-complete for DL-Lite $_{\text{RDFS}}$. For \mathcal{ELH} , the problem is P-complete in data complexity and PSpace-complete in combined complexity. All lower bounds hold also for CRPQs and in the absence of role inclusions.*

We split the proof of this theorem into parts, with the lower bounds shown in the next section, and the (more involved) proofs of the upper bounds outlined in Section 5.

4 Lower Bounds

We start by establishing the required lower bounds.

Proposition 1. *Boolean CRPQ entailment is*

1. NLSpace-hard in data complexity for DL-Lite $_{\text{RDFS}}$;
2. P-hard in data complexity for \mathcal{EL} ;
3. NP-hard in combined complexity for DL-Lite $_{\text{RDFS}}$;
4. PSpace-hard in combined complexity for DL-Lite and \mathcal{EL} .

Proof. Statement (1) follows from the analogous result for graph databases [8]. It can be shown by a simple reduction from the NLSpace-complete directed reachability problem: y is reachable from x in a directed graph G if and only if (x, y) is an answer to $r^*(x, y)$ w.r.t. the ABox \mathcal{A}_G encoding G . Statement (2) is immediate given the P-hardness in data complexity of CQ entailment in \mathcal{EL} [7], and (3) follows from the well-known NP-hardness in combined complexity of CQ entailment for databases [1].

For statement (4), we give a reduction from the problem of emptiness of the intersection of an arbitrary number of regular languages, which is known to PSpace-complete [10]. Consider some regular languages L_1, \dots, L_n over alphabet Σ . We will use the symbols in Σ as role names, and we add a concept name A . Then we set $\mathcal{A} = \{A(a)\}$ and $q = \exists x L_1(a, x) \wedge \dots \wedge L_n(a, x)$. For DL-Lite, we will use the

following TBox: $\mathcal{T} = \{A \sqsubseteq \exists r \mid r \in \Sigma\} \cup \{\exists r^- \sqsubseteq \exists s \mid r, s \in \Sigma\}$. For \mathcal{EL} , we can use $\mathcal{T} = \{A \sqsubseteq \exists r.A \mid r \in \Sigma\}$. Notice that in both cases the canonical model $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ consists of an infinite tree rooted at a such that every element in the interpretation has a unique r -child for each $r \in \Sigma$ (and no other children). Thus, we can associate to every domain element the word over Σ given by the unique path from a , and moreover, for every word $w \in \Sigma^*$ we can find an element e_w whose path from a is exactly w . This means that if $w \in L_1 \cap \dots \cap L_n$, we obtain a match for q in the canonical model by mapping x to e_w . Conversely, if q is entailed, then any match in the canonical model defines a word which belongs to every L_i , which means $L_1 \cap \dots \cap L_n$ is non-empty. \square

5 Upper Bounds

The main objective of this section will be to define procedure for deciding $\mathcal{I}_{\mathcal{T}, \mathcal{A}} \models q$ for a given KB \mathcal{T}, \mathcal{A} and a given ℓ -C2RPQ q . The procedure comprises two main steps. First, we rewrite q into a set Q of ℓ -C2RPQs such that $\mathcal{I}_{\mathcal{T}, \mathcal{A}} \models q$ if and only if $\mathcal{I}_{\mathcal{T}, \mathcal{A}} \models q'$ for some $q' \in Q$. The advantage of the rewritten queries is that in order to decide whether $\mathcal{I}_{\mathcal{T}, \mathcal{A}} \models q'$, we will only need to consider matches which map the variables to $\text{Ind}(\mathcal{A})$. The second step evaluates the rewritten queries over the core part of the canonical model involving only $\text{Ind}(\mathcal{A})$.

Preliminary Notions In order to more easily manipulate regular languages, it will prove convenient to use NFAs rather than regular expressions. Thus, in what follows, we assume all binary atoms take the form $\alpha(t, t')$, where α is an NFA over $\mathcal{B}(\overline{\mathbf{N}}_{\mathbf{R}}) \times \mathcal{B}(\mathbf{N}_{\mathbf{C}})$. Given $\alpha = \langle S, \Sigma, \delta, s_0, F \rangle$, we use $\alpha_{s, G}$ to denote the NFA $\langle S, \Sigma, \delta, s, G \rangle$, i.e. the NFA with the same states and transitions as α but with initial state s and final states G .

A key to defining our rewriting procedure will be to understand how an atom $L(t, t')$ can be satisfied in the anonymous part of the canonical model $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$. A subtlety arises from the fact that the path witnessing the satisfaction of an atom $L(t, t')$ may be quite complicated: it may move both up and down, passing by the same element multiple times, and possibly descending below t' . This will lead us to decompose an atom $L(t, t')$ into multiple “smaller” atoms corresponding to segments of the L -path which are situated wholly above or below an element. Importantly, we know that the canonical model displays a high degree of regularity, since whenever two elements p_1 and p_2 in the anonymous part end with the same concept (i.e. $\text{Tail}(p_1) = \text{Tail}(p_2)$), the submodels $\mathcal{I}_{\mathcal{T}, \mathcal{A}}|_{p_1}$ and $\mathcal{I}_{\mathcal{T}, \mathcal{A}}|_{p_2}$ are isomorphic. In particular, this means that if $\text{Tail}(p_1) = \text{Tail}(p_2)$, then p_1 is an L -successor of itself in the interpretation $\mathcal{I}_{\mathcal{A}, \mathcal{T}}|_{p_1}$ just in the case that p_2 is an L -successor of itself in the interpretation $\mathcal{I}_{\mathcal{A}, \mathcal{T}}|_{p_2}$.

We now wish to define a way of testing for a given TBox \mathcal{T} and NFA α with states s, s' whether $\text{Tail}(e) = C$ ensures that there is a loop from e back to itself, situated wholly within $\mathcal{I}_{\mathcal{T}, \mathcal{A}}|_e$, which takes α from state s to state s' . To this end, we construct a table Loop_{α} which contains for each pair s, s' of states in α , a subset of BC. If \mathcal{T} is a DL-Lite $_{\mathcal{R}}$ TBox, then Loop_{α} is defined inductively using the following rules:

- (1) for every $s \in S$, $\text{Loop}_{\alpha}[s, s] = \text{BC}$
- (2) if $C \in \text{Loop}_{\alpha}[s_1, s_2]$ and $C \in \text{Loop}_{\alpha}[s_2, s_3]$, then $C \in \text{Loop}_{\alpha}[s_1, s_3]$

- (3) if $C \in \text{BC}$, $\mathcal{T} \models C \sqsubseteq \exists R, \exists R^- \in \text{Loop}_\alpha[s_2, s_3]$,
 $(s_1, (\Gamma, \Upsilon), s_2) \in \delta$, $(s_3, (\Gamma', \Upsilon'), s_4) \in \delta$,
 $\{U \in \overline{\text{N}}_{\text{R}} \mid \mathcal{T} \models R \sqsubseteq U\} \models \Gamma$, $\{A \in \text{N}_{\text{C}} \mid \mathcal{T} \models \exists R^- \sqsubseteq A\} \models \Upsilon$,
 $\{U \in \overline{\text{N}}_{\text{R}} \mid \mathcal{T} \models R^- \sqsubseteq U\} \models \Gamma'$, and $\{A \in \text{N}_{\text{C}} \mid \mathcal{T} \models C \sqsubseteq A\} \models \Upsilon'$,
then $C \in \text{Loop}_\alpha[s_1, s_4]$

For \mathcal{ELH} , we replace the third rule by:

- (3') if $C \in \text{BC}$, $\mathcal{T} \models C \sqsubseteq \exists r.D$, $D \in \text{Loop}_\alpha[s_2, s_3]$,
 $(s_1, (\Gamma, \Upsilon), s_2) \in \delta$, $(s_3, (\Gamma', \Upsilon'), s_4) \in \delta$,
 $\{s \in \overline{\text{N}}_{\text{R}} \mid \mathcal{T} \models r \sqsubseteq s\} \models \Gamma$, $\{A \in \text{N}_{\text{C}} \mid \mathcal{T} \models D \sqsubseteq A\} \models \Upsilon$,
 $\{s^- \in \overline{\text{N}}_{\text{R}} \mid \mathcal{T} \models r \sqsubseteq s\} \models \Gamma'$, and $\{A \in \text{N}_{\text{C}} \mid \mathcal{T} \models C \sqsubseteq A\} \models \Upsilon'$,
then $C \in \text{Loop}_\alpha[s_1, s_4]$

Note that the table Loop_α can be constructed in polynomial time in $|\mathcal{T}|$ and $|\alpha|$ since entailment of inclusions is polynomial for both $\text{DL-Lite}_{\mathcal{R}}$ and \mathcal{ELH} . The following lemma shows that Loop_α has the desired meaning:

Lemma 2. *For every element $p \in \Delta^{\mathcal{I}_{\mathcal{A}, \mathcal{T}}} \setminus \text{Ind}(\mathcal{A})$: $\text{Tail}(p) \in \text{Loop}_\alpha[s, s']$ if and only if p is an $L(\alpha_{s, s'})$ -successor of itself in the interpretation $\mathcal{I}_{\mathcal{A}, \mathcal{T}}|_p$.*

Query Rewriting Our aim is to rewrite our query in such a way that we do not need to map any variables to the anonymous part of the model. We draw our inspiration from a query rewriting procedure for Horn- SHIQ described in [9]. The main intuition is as follows. Suppose we have a match π for q which maps some variable y to the anonymous part, and no other variable is mapped below $\pi(y)$. Then we modify q so that it has essentially the same match except that variables mapped to $\pi(y)$ are now mapped to the (unique) parent of $\pi(y)$ in $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$. The delicate point is that we must “split” atoms of the form $\alpha(t, t')$ with $y \in \{t, t'\}$ into the parts which are satisfied in the subtree $\mathcal{I}_{\mathcal{T}, \mathcal{A}}|_{\pi(y)}$ (these have already been shown to hold, so can be dropped), and those which occur above $\pi(y)$, whose satisfaction still needs to be determined and thus must be incorporated into the new query. With each iteration of the rewriting procedure, we obtain a query which has a match which maps variables “closer” to the core of $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$, until eventually we find some query that has a match which maps all terms to $\text{Ind}(\mathcal{A})$.

We now give a recursive non-deterministic query rewriting procedure which implements the above intuition.

PROCEDURE $\text{rewrite}(q, \mathcal{T})$

1. Choose either to output q or to continue.
2. Choose a non-empty set $\text{Leaf} \subseteq \text{vars}(q)$ and $y \in \text{Leaf}$. Rename all variables in Leaf to y .
3. Choose some concept $C \in \text{BC}$ such that $\{B \mid \mathcal{T} \models C \sqsubseteq B\} \models \varphi$ for every atom $\varphi(y)$. Drop all such atoms from q .
4. For each atom $\alpha(t, t')$ where $\alpha = \langle S, \Sigma, \delta, s, F \rangle$ is a NFA and $y \in \{t, t'\}$,
 - choose a sequence s_1, \dots, s_n of distinct states from S such that $s_n \in F$,
 - replace the atom $\alpha(t, t')$ in q by the atoms $\alpha_{s, s_1}(t, y)$, $\alpha_{s_1, s_2}(y, y)$, \dots , $\alpha_{s_{n-2}, s_{n-1}}(y, y)$, $\alpha_{s_{n-1}, s_n}(y, t')$.

5. Drop all atoms $\alpha_{s,s'}(y, y)$ such that $C \in \text{Loop}_\alpha[s, s']$.
6. Choose some $D \in \text{BC}$ and $R \in \overline{\text{NR}}$ such that:
 - (a) if \mathcal{T} is a DL-Lite \mathcal{R} TBox, then $C = \exists R^-$ and $\mathcal{T} \models D \sqsubseteq \exists R$.
 - (a') if \mathcal{T} is an \mathcal{ELH} TBox, then $R \in \text{NR}$ and $\mathcal{T} \models D \sqsubseteq \exists R.C$.
 - (b) for each atom of the form $\alpha(y, x)$ with $\alpha = \langle S, \Sigma, \delta, s, F \rangle$, there exist $s' \in S$ and $(\Gamma, \Upsilon) \in \Sigma$ with $(s, (\Gamma, \Upsilon), s') \in \delta$ and $\{U \in \overline{\text{NR}} \mid \mathcal{T} \models R^- \sqsubseteq U\} \models \Gamma$.
 - (c) for each atom of the form $\alpha(x, y)$ with $\alpha = \langle S, \Sigma, \delta, s, F \rangle$, there exists $s'' \in S$, $s_f \in F$ and some $(\Omega, \Theta) \in \Sigma$ with $(s'', (\Omega, \Theta), s_f) \in \delta$ such that $\{U \in \overline{\text{NR}} \mid \mathcal{T} \models R \sqsubseteq U\} \models \Omega$ and $\{A \in \text{NC} \mid \mathcal{T} \models C \sqsubseteq A\} \models \Theta$.
 Note that both (b) and (c) apply to an atom of the form $\alpha(y, y)$.
7. Replace
 - each atom $\alpha(y, x)$ with $x \neq y$ by atoms $\alpha_{s',s_f}(y, x)$ and $\Upsilon(y)$
 - each atom $\alpha(x, y)$ with $x \neq y$ by $\alpha_{s,s''}(x, y)$
 - each atom $\alpha(y, y)$ by atoms $\alpha_{s',s''}(y, y)$ and $\Upsilon(y)$
 with $s, s', s'', s_f, \Upsilon$ as in Step 6.
8. If $D \in \text{NC}$ is the concept chosen in Step 6, add $D(y)$ to q . If $D = \exists P^-$, add $\alpha_P(z, y)$ to q , where z is a fresh variable and $L(\alpha_P) = \{(P, \text{true})\}$. Go to Step 1.

Slightly abusing notation, we will use $\text{rewrite}(q, \mathcal{T})$ to denote the set of queries which are output by some execution of rewrite on input q, \mathcal{T} . We remark that the number of variables and atoms in each query in $\text{rewrite}(q, \mathcal{T})$ is linearly bounded by the original q . This is the key property used to show the following:

Lemma 3. *There are only exponentially many queries in $\text{rewrite}(q, \mathcal{T})$ (up to equivalence), and each one has size polynomial in $|q|$.*

The next lemma shows that using $\text{rewrite}(q, \mathcal{T})$, we can reduce the problem of finding an arbitrary query match to finding a match involving only ABox individuals.

Lemma 4. *$\mathcal{T}, \mathcal{A} \models q$ if and only if there exists a match π for some query $q' \in \text{rewrite}(q, \mathcal{T})$ in $\mathcal{I}_{\mathcal{A}, \mathcal{T}}$ such that $\pi(t) \in \text{Ind}(\mathcal{A})$ for every term t in q' .*

Query Evaluation We have reduced $\mathcal{T}, \mathcal{A} \models q$ to checking whether there is a match π for some $q' \in \text{rewrite}(q, \mathcal{T})$ with $\pi(t) \in \text{Ind}(\mathcal{A})$ for every term t in q' . However, even when all terms are mapped to ABox individuals, the paths between them may need to pass by the anonymous part in order to satisfy the regular expressions in the query. To handle this problem, we define a relaxed notion of query entailment, which exploits the fact that if all variables are mapped to $\text{Ind}(\mathcal{A})$, only loops (that is, paths from an individual a to itself in $\mathcal{I}_{\mathcal{A}, \mathcal{T}}|_a$) may participate in the paths between them. Hence, we look for paths in the ABox that may use such loops to skip states in the query automata.

Thus, as part of our evaluation procedure, we will need to decide for a given individual a whether a is an $L(\alpha_{s,s'})$ -successor of itself in $\mathcal{I}_{\mathcal{A}, \mathcal{T}}|_a$. We cannot use the table Loop_α directly, since it does not take into account the concepts which are entailed due to ABox assertions. We note however that the set of loops starting from a given individual is fully determined by the set of concepts in BC which the individual satisfies. We thus introduce a new table ALoop_α such that $\text{ALoop}_\alpha[s, s']$ contains all subsets $G \subseteq \text{BC}$ such that a is an $L(\alpha_{s,s'})$ -successor of itself in $\mathcal{I}_{\mathcal{A}, \mathcal{T}}|_a$ whenever $G = \{C \in \text{BC} \mid a \in C^{\mathcal{I}_{\mathcal{A}, \mathcal{T}}}\}$. Note that the size of ALoop_α is exponential in $|\mathcal{T}|$, but the associated decision problem is in P:

Lemma 5. *It can be decided in polytime in $|\mathcal{T}|$ and $|\alpha|$ whether $G \in \text{ALoop}_\alpha[s, s']$.*

Definition 2. *We write $\mathcal{T}, \mathcal{A} \approx q$ if there is a mapping π from the terms in q to $\text{Ind}(\mathcal{A})$ such that:*

- (a) $\pi(c) = c$ for each $c \in \mathbb{N}_I$,
- (b) $\{A \in \mathbb{N}_C \mid \mathcal{T}, \mathcal{A} \models A(\pi(t))\} \models \beta$ for each atom $\beta(t)$ in q , and
- (c) for each $\alpha(t, t') \in q$ with $\alpha = \langle S, \Sigma, \delta, s, F \rangle$, there is a sequence $(a_0, s_0), \dots, (a_n, s_n)$ of distinct pairs from $\text{Ind}(\mathcal{A}) \times S$ such that $a_0 = \pi(t)$, $a_n = \pi(t')$, $s_0 = s$, $s_n \in F$, and for every $0 \leq i < n$, one of the following holds:
 - (i) $a_i = a_{i+1}$ and $\{C \in \text{BC} \mid \mathcal{T}, \mathcal{A} \models C(a_i)\} \in \text{ALoop}_\alpha[s_i, s_{i+1}]$
 - (ii) there is some $(\Gamma, \Upsilon) \in \Sigma$ with $(s_i, (\Gamma, \Upsilon), s_{i+1})$, $\{R \in \overline{\mathbb{N}}_R \mid \mathcal{T}, \mathcal{A} \models R(a_i, a_{i+1})\} \models \Gamma$ and $\{A \in \mathbb{N}_C \mid \mathcal{T}, \mathcal{A} \models A(a_{i+1})\} \models \Upsilon$.

Lemma 6. $\mathcal{T}, \mathcal{A} \models q$ if and only if $\mathcal{T}, \mathcal{A} \approx q'$ for some $q' \in \text{rewrite}(q, \mathcal{T})$.

Using the preceding lemma, we can derive our upper bounds:

Proposition 2. *Boolean ℓ -C2RPQ entailment is*

1. NLSpace in data complexity for DL-Lite $_{\mathcal{R}}$ and DL-Lite $_{\text{RDFS}}$;
2. P in data complexity for \mathcal{ELH} ;
3. NP in combined complexity for DL-Lite $_{\text{RDFS}}$;
4. PSPACE in combined complexity for DL-Lite $_{\mathcal{R}}$ and \mathcal{ELH} .

Proof. By Lemmas 4 and 6, we can reduce $\mathcal{T}, \mathcal{A} \models q$ to deciding whether $\mathcal{T}, \mathcal{A} \approx q'$ for some $q' \in \text{rewrite}(q, \mathcal{T})$. For items 1 and 2, if \mathcal{T} and q are fixed, then computing $\text{rewrite}(q, \mathcal{T})$ requires only constant time in $|\mathcal{A}|$. To decide whether $\mathcal{T}, \mathcal{A} \approx q'$ for $q' \in \text{rewrite}(q, \mathcal{T})$, we guess a mapping π from the terms in q' to $\text{Ind}(\mathcal{A})$ and verify that it satisfies the conditions in Definition 2. Note that for condition (c), we cannot keep the whole sequence $(a_0, s_0), \dots, (a_n, s_n)$ in memory at once, so we use a binary counter that counts up to $\text{Ind}(\mathcal{A}) \times |S|$ and store only one pair of nodes $(a_i, s_i), (a_{i+1}, s_{i+1})$ at a time. To verify conditions (b) and (c)(ii) we need checks of the form $\{R \in \overline{\mathbb{N}}_R \mid \mathcal{T}, \mathcal{A} \models R(a, b)\} \models \Gamma$ and $\{A \in \mathbb{N}_C \mid \mathcal{T}, \mathcal{A} \models A(a)\} \models \Upsilon$. Each one amounts to a fixed number of instance checks (one for each symbol in Γ or Υ), hence the data complexity of these checks is the same as for instance checking in the corresponding DL: in AC_0 for DL-Lite $_{\mathcal{R}}$, and in P for \mathcal{ELH} . This yields the desired upper bounds: NLSpace for the former, and $\text{NLSpace}^{\text{P}} = \text{P}$ for the latter.

For statement 4, instead of building the whole set $\text{rewrite}(q, \mathcal{T})$, which can be exponential, we generate a single $q' \in \text{rewrite}(q, \mathcal{T})$ non-deterministically. More precisely, we take the initial query q and apply a sequence of rewriting steps to obtain some $q' \in \text{rewrite}(q, \mathcal{T})$. By Lemma 3, every query in $\text{rewrite}(q, \mathcal{T})$ can be generated after at most exponentially many steps, so we can use a polynomial-sized counter to check when we have reached this limit. Since each rewritten query is of polynomial size (Lemma 3), and we keep a single query in memory at a time, the generation of a single query in $\text{rewrite}(q, \mathcal{T})$ requires only polynomial space. Then we can use the same strategy as above to decide in polynomial space whether $\mathcal{T}, \mathcal{A} \approx q'$. We thus

have a non-deterministic polynomial space procedure for deciding $\mathcal{T}, \mathcal{A} \models q$. Using the well-known fact that $\text{NPSpace} = \text{PSpace}$, we obtain the desired upper bound.

For statement 3, we note that if \mathcal{T} is an DL-Lite_{RDFS} TBox, then the query cannot be rewritten, i.e. $\text{rewrite}(q, \mathcal{T}) = \{q\}$. Thus, it suffices to decide whether $\mathcal{T}, \mathcal{A} \models q$. We then remark that the procedure described above is in NP, since we guess a (polysize) mapping π and verify in polytime that π satisfies the conditions of Definition 2. \square

6 Future Work

In future work, we plan to study what types of restrictions on the TBox and query lead to better combined complexity. We also wish to explore other types of path-based query languages. One interesting extension which has been recently proposed for graph databases [3] is the addition of *path variables*. In Boolean queries, these prove useful for speaking about equality of paths. For example, one might want to find whether there is a chain of advisors of the same length between both Edmund Clarke and Bernoulli, and Jack Minker and Bernoulli. Things become even more interesting if we allow path variables in the output. By outputting the paths for the preceding query, we could discover that Clarke and Minker have a path to Bernoulli of length 12. We could even output the common areas of expertise of the scientists along the path to find out that both have an 8-step path to some physicist (Poisson and Fourier), that in turn reach Bernoulli via an important figure in analysis (Lagrange) and a graph theorist (Euler).

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proc. of IJCAI. pp. 364–369 (2005)
3. Barceló, P., Hurtado, C.A., Libkin, L., Wood, P.T.: Expressive languages for path queries over graph-structured data. In: Proc. of PODS. pp. 3–14 (2010)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
5. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proc. of AAAI. pp. 391–396 (2007)
6. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Proc. of IJCAI. pp. 714–720 (2009)
7. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR. pp. 260–270 (2006)
8. Consens, M.P., Mendelzon, A.O.: Graphlog: a visual formalism for real life recursion. In: Proc. of PODS. pp. 404–416 (1990)
9. Eiter, T., Ortiz, M., Šimkus, M., Tran, T., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of AAAI (2012)
10. Kozen, D.: Lower bounds for natural proof systems. In: Proc. of FOCS. pp. 254–266 (1977)
11. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the horn fragments of the description logics SHOIQ and SROIQ. In: Proc. of IJCAI. pp. 1039–1044 (2011)

Experiences in Mapping the Business Intelligence Model to Description Logics, and the Case for Parametric Concepts

Alexander Borgida¹, Jennifer Horkoff², John Mylopoulos² and Riccardo Rosati³

¹ Dept. of Computer Science, Rutgers University, NJ, USA

² Dept. of Computer Science, Univ. of Toronto, Canada

³ DIS, Sapienza Università di Roma, Italy

1 Summary

BIM is a new language for capturing business models containing information relevant for strategic analysis of business operations. It has been used in several large case studies and is being pursued in industry.

The paper introduces the key notions of BIM, including goals, evidence, and influence. It also outlines their translation into DL axioms, forming an upper-level ontology. Specific BIM domain models then result from the addition of axioms to this. The result provides both a formal semantics of the BIM language, and all the familiar advantages of decidable DL reasoning, including consistency checking, defined-concept classification, and, in our case “What if” scenario analysis. We focus on the parts of the translation which are most interesting, including: i) modeling “evidence and pursuit propagation” about goals, ii) dealing with “meta-properties”, which were introduced as a result of an ontological analysis of previous BI languages, and iii) the repeated need for too many similar axioms.

For the last two, we sketch how **parametrized concepts**, together with rules, would significantly help knowledge-base maintenance. This opens up a new research area in hybrid DL+rule KBs, involving rules that generate new axioms.

2 Introduction to BIM

Business intelligence (BI) offers considerable potential for gaining insights into day-to-day business operations, as well as longer term opportunities and threats. Most businesses have a significant investment in BI; however, much of the information is data oriented – mostly low-level values difficult to understand in terms of business strategy. Instead, there is a need for analysis using terms like strategic objectives, business models, processes, markets, trends and risks.

Several BI modeling techniques exist already: the Business Motivation Model (BMM) [1], Strategy Maps (SM) [2], Balanced Scorecards (BSc) [3], Goal Modeling frameworks (GM) [4, 5]. These languages introduce many concepts informally, making it difficult to distinguish between them; (except for GM), do not support reasoning over models; and do not offer facilities for standard conceptual modeling of domain entities. The Business Intelligence Model (BIM) was introduced [8, 7] to rationalize and extend notions in previous languages.

A portion of a realistic BIM schema for the credit card industry is shown in in Fig. 1, using a (provisional) graphical notation based on the i* GM [4].

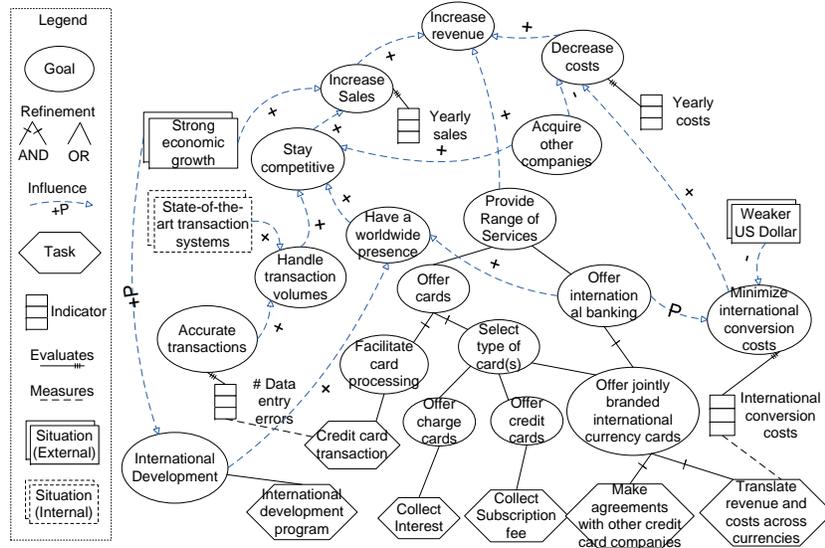


Figure 1: Fragment of BIM schema for banking industry

Generally, the model shows the decomposition of basic business services (e.g., offer cards, offer international banking) into operational tasks, their effects on strategic goals, an assessment of influencing situations, and measurement through indicators. So BIM focuses on four types of things: **Situations**, **Indicators**, **Tasks** and **Entities**, which are subclasses of **BIM_Thing**. (We abbreviate this to Thing, using OWL:Thing, if we need, to talk about the top DL concept.) Situations are partial descriptions of world state, which may affect business objectives, and in BIM are specialized into **Goals**, **Operational Situations**, and **Domain Assumptions**.

Goals are situations that may be desired by the modeling organization, such as “Increase revenue”. Goals may or may not be actively *pursued* at any time, and have an *evidence* value. As usual in GM, goals are refined until one finds actions that help achieve them (Tasks), or domain assertions that are assumed to hold in the real world.

Additional concepts found in other BI languages can be obtained in BIM by providing values to meta-attributes (see Section 3.4). These provide an optional richer subclass structure for BIM without over-complicating the initial model design, and language learning.

BIM includes five different types of relationships between things: **influences**, **evaluates**, **refines**, and **measures**, but their domains/ranges are restricted to various subclasses of Thing.

We are only going to touch on those aspects of BIM that raise interesting issues for DL modeling. Unfortunately, this excludes one of the most interesting features of BIM: the notion of *indicators*, which measure the performance of some business activity. These use heavily numeric functions [9], and we were unable to model them in depth in OWL2.

3 The Translation of BIM to DL

3.1 Some BIM Classes and their Axioms

The DL axioms capturing the semantics⁴ of the subclass hierarchy under Thing are standard, as is the specification of disjointness between sibling classes.

BIM allows accumulation of *evidence* for or against every thing in BIM. The question “Evidence for . . . ?” is answered depending on the specific type of thing. So BIM tracks evidence for the *occurrence* of situations, the *satisfaction* of goals and domain assumptions, the *performance* of indicators, the *execution* of tasks, and the *existence* of entities. In this way, we use BIM to monitor the state of relevant business concepts. Following [6], we will accumulate various *qualitative* kinds of evidence (for and against) from *multiple sources*, and combine them using a multi-valued logic approach, so that the value of *evidence* can be zero or more of StrongEvidenceFor (SF), WeakEvidenceFor (WF), StrongEvidenceAgainst (SA), and WeakEvidenceAgainst (WA). We therefore have

$$\text{evidence} \sqsubseteq_r \text{Thing} \times \{\text{SF}, \text{WF}, \text{WA}, \text{SA}\}$$

Rather than constantly asking whether the evidence for something is strong by checking subsumption by ($\text{evidence} : \text{SF}$) (a synonym for $\exists \text{evidence}.\{\text{SF}\}$), we will find it convenient to define four concepts like

$$\text{SFThing} \equiv \text{Thing} \sqcap (\text{evidence} : \text{SF})$$

Such repetitions of axioms are frequent and annoying for BIM so we introduce schemas for them, using the notation of programming language features such as C++ templates and Java generics:

$$\text{Thing}\langle ?V \rangle \equiv \text{Thing} \sqcap (\text{evidence} : ?V) \text{ for } ?V \in \{\text{SF}, \text{WF}, \text{WA}, \text{SA}\}$$

Note that in C++ and Java, parameters may be restricted to be subtypes of other types; for example, a class `SortedList<?V>` may require `?V` to be a subtype of `Comparable` to guarantee a `lessThan` operation. We will present parameterized DL concepts using rules, whose body limits the parameter values using *P-facts* (think of these as “parameter facts”), about individuals or “punned” class identifiers. Thus after introducing P-predicate *EvidValues*, with instances SF, WF, WA and SA, we can write:

⁴ We have intentionally not chosen a particularly restrictive DL at this point, since a lower bound on complexity of BIM reasoning can only be obtained directly. Since we want to have off-the-shelf reasoners, we have limited ourselves to OWL2, though nominals, transitivity and even inverses are not strictly needed for reasoning.

$$\text{Goal}\langle ?V \rangle \equiv \text{Goal} \sqcap (\text{evidence} : ?V) :- \text{EvidValues}\langle ?V \rangle$$

For more complicated cases, and more uniform notation, we might prefer to use rules extending the syntax of higher-order logics such as $\text{Hi}(\mathcal{D})$ [11]:

$$\begin{aligned} \text{Define}_C(\text{Goal}\langle ?V \rangle, \text{And}(\text{Goal}, \text{HasValue}(\text{evidence}, ?V))) :- \\ \text{InstanceOf}_C\langle ?V, \text{EvidValues} \rangle \end{aligned}$$

Returning to BIM, we need to also say that strong implies weak evidence, using axioms:

$$\text{Thing}\langle \text{SF} \rangle \sqsubseteq \text{Thing}\langle \text{WF} \rangle \quad \text{Thing}\langle \text{SA} \rangle \sqsubseteq \text{Thing}\langle \text{WA} \rangle$$

Information about the *pursuit* property can also come from multiple situations, so its value is a subset of $\{Pur, NegPur\}$.

3.2 BIM Relationships: Their Domains and Ranges

Influences. The *influences* relationship is used to represent the (transmission of) (un)favorable effects on situations. As natural, we represent BIM relationships by DL properties, so we have simply

$$\text{influences} \sqsubseteq_r \text{Situation} \times \text{Situation} \quad \text{infBy} \equiv_r \text{influences}^-$$

Borrowing from GM [6], there are a variety of influence links: a ++ (resp. +) represents strong (resp. partial) positive influence on *evidence*, and a --/- influence link represents strong/partial negative one. In Fig. 1, “Strong economic growth” has a partial positive influence on “Increase sales”. Influence links also can affect the *pursuit* of goals, using optional influence annotations P and !P, representing pursuing and its denial respectively. The different kinds of labels on influence will be encoded thru sub-properties of *influences*, with the original labels as prefixes separated by an underscore. Thus “StayCompetitive” positively influences “IncreaseSales” from Fig. 1, would be encoded, in part, by the axiom

$$\text{IncreaseSales} \sqsubseteq \exists +_infBy.\text{StayCompetitive}$$

Refines. The *refines* relationship helps decompose concepts into other, often more detailed, concepts of that type. *Refines* is also used to determine evidence for/against a thing, based on the *evidence* for/against its refinements. Unlike other relationships (or UML associations), *refines* is limited to different pairs of sub-(domain,range) pairs. Thus a goal refines other goals (not other kinds of situations), but goals can be refined into goals, domain assumptions (DA) or tasks:

$$\text{Goal} \sqsubseteq \forall \text{refines}.\text{Goal} \quad \exists \text{refines}.\text{Goal} \sqsubseteq (\text{Goal} \sqcup \text{DA} \sqcup \text{Task})$$

Every other subclass of Situation, except Task, only has axioms like

$$\text{Situation} \sqsubseteq \forall \text{refines}.\text{Situation} \quad \exists \text{refines}.\text{Situation} \sqsubseteq \text{Situation}$$

Refinements are by default interpreted disjunctively, but can also be marked as explicitly AND-ed: e.g., both “Facilitate card processing” and “Select type of card(s)” are required to satisfy “Offer cards”. Since on any particular node, we want all refinements to be AND-ed or OR-ed, we add a subclass AND_Thing of Thing, and have axioms:

$$\text{AND_Thing} \sqsubseteq \text{Thing} \quad \text{OR_Thing} \equiv \text{Thing} \sqcap \neg \text{AND_Thing}$$

These concepts will be used below to define the propagation of evidence values for AND and OR refinements.

3.3 Evidence and Pursuit

Recall that each BIM thing has an *evidence* property, with value a subset of {SF, WF, WA, SA} and *pursuit* property, with value a subset of {Pur, NegPur}. We provide the precise rules for relating both *evidence* and *pursuit* values in the presence of *refines* and *influences* relationships between nodes.

For *refines*, we use the rules for combining evidence on AND and OR nodes inspired from [6]. Positive evidence values from the sources are propagated to the target according to its node kind: on an OR node, it is enough to have one refiner with $V=SF,WF$ to get V ; on an AND node, all refiners must have V :

$$\begin{aligned} \text{OR_Thing} \sqcap \exists \text{refinedBy.Thing}\langle?V\rangle &\sqsubseteq \text{Thing}\langle?V\rangle \quad :- ?V \in \{SF,WF\} \\ \text{AND_Thing} \sqcap \forall \text{refinedBy.Thing}\langle?V\rangle &\sqsubseteq \text{Thing}\langle?V\rangle \quad :- ?V \in \{SF,WF\} \end{aligned}$$

For negative evidence, the converse holds:

$$\begin{aligned} \text{OR_Thing} \sqcap \forall \text{refinedBy.Thing}\langle?V\rangle &\sqsubseteq \text{Thing}\langle?V\rangle \quad :- ?V \in \{SA,WA\} \\ \text{AND_Thing} \sqcap \exists \text{refinedBy.Thing}\langle?V\rangle &\sqsubseteq \text{Thing}\langle?V\rangle \quad :- ?V \in \{SA,WA\} \end{aligned}$$

Note that the presence of common DL concept constructors, such as qualified number restrictions $\geq n R.C$, immediately suggests extending BIM to support AND(n) nodes, which require the satisfaction of at least n refinements.

For *influences*, ideally we would separate the evidence and pursuit aspects, but the desired semantics sometimes requires knowing both aspects at the same time. So we introduce a taxonomy of properties, starting with leafs like $++P_InfBy$. These are then grouped in various ways as subproperties of others like $InfByP$ and $infBy++$; in turn, $infBy++$ and $infBy+$ are subproperties of $infByPositively$. This allows us to later state some axioms once, for a higher property, rather than repeat it for each subproperty.

The evidence values of the source are propagated to the target depending on the strength of the source evidence and the influence label. The 12 rules from [6] could then be written as 12 axioms, such as

$$\exists \text{infBy}++. \text{Goal}\langle SF \rangle \sqsubseteq \text{Goal}\langle SF \rangle$$

However, if we used parametrized classes and axioms, the following shows much more intuitively what happens in the 6 axioms involving positive influence:

$$\exists \text{infByPositively.} \text{Goal}\langle?V\rangle \sqsubseteq \text{Goal}\langle?V\rangle \quad :- \text{EvidValues}\langle?V\rangle$$

The meaning of negative influences requires more complexity, because the “polarity” of the evidence value must be switched. Using P-facts $\text{complement}(SF,WF)$ and $\text{complement}(SF,WF)$, with symmetric *complement*, we encode the 6 other axioms with the rule

$$\exists \text{infByNegatively.} \text{Goal}\langle?V\rangle \sqsubseteq \text{Goal}\langle?W\rangle \quad :- \text{complement}\langle?V,?W\rangle.$$

The propagation of pursuit along influence links from goals to goals is similar to that of evidence with *Pur*, *NotPur*, *P*, *!P* playing the role of $++$, $--$, *SF*, *SA* respectively. Again, we can choose 4 ordinary axioms, or 2 parameterized ones.

Reasoning with Missing Pursuit. Recall that only goals have a *pursuit* attribute. This means complex special cases.

If the source does not have a *pursuit* attribute, e.g. a situation influencing a goal, the satisfaction polarity of the source determines the pursuit of the target in P and !P influence types. This is one of the 4 axioms for this:

$$\exists \text{InfByP} . (\neg \text{Goal} \sqcap (\text{Thing}\langle \text{SF} \rangle \sqcup \text{Thing}\langle \text{WF} \rangle)) \sqsubseteq \text{PurGoal}$$

If the source has a *pursuit* attribute but the destination does not, e.g. a goal influencing a situation, then the influence of the source *evidence* on the destination *evidence* only occurs when the goal is pursued, in case P is on the label, or not pursued, in the case of !P. For example, if the goal is satisfied and pursued, and the label is +P, the target situation partially occurs. If the goal is satisfied and not pursued, the situation has no incoming evidence from that goal. This requires replacing each of the original 12 axioms for propagating *evidence* by pairs such as

$$\exists ++\text{P_infBy} . \text{Goal} \sqcap \text{PurGoal} \sqsubseteq \text{SFThing}$$

$$\exists ++\text{!P_infBy} . \text{SFGGoal} \sqcap \text{NotPurGoal} \sqsubseteq \text{SFThing}$$

which check the appropriate combination of edge and node labels. Again, these could be stated much more succinctly by using parametric concepts and properties:

$$\exists \text{infBy}\langle ?S, ?P \rangle . (\text{EGoal}\langle ?S \rangle \sqcap \text{PGoal}\langle ?P \rangle) \sqsubseteq \text{EThing}\langle ?S \rangle \quad :- \\ \text{EvidValue}\langle ?S \rangle, \text{PursuitValue}\langle ?P \rangle.$$

where we must now distinguish parameterizing concepts like Goal and Thing by *evidence* or by *pursuit*

$$\text{EGoal}\langle ?V \rangle \equiv \text{Goal} \sqcap (\text{evidence} : ?V)$$

$$\text{PGoal}\langle ?V \rangle \equiv \text{Goal} \sqcap (\text{pursuit} : ?V)$$

Translating BIM Models to DL Given a specific model, such as the one in Fig.1, we need to generate axioms that connect it to the generic terms of BIM axiomatized above. For this, we make every node a class, and add axioms describing its “BIM type”. For example, node “Offer International Banking”, which is a goal, would generate:

$$\text{OfferInternationalBanking} \sqsubseteq \text{Goal} \sqcap \text{AND_Thing}$$

We also add axioms declaring the disjointness of all nodes. Finally, every edge is translated into DL axioms in a manner that respects the following intuition of GM users: for every instance of a top-level goal, there is a separate set of instances connected to it, which result in an isomorphic copy of the (concept level) graph. This is assured by pairs of axioms, illustrated for the + influences edge from StayCompetitive to IncreaseSales:

$$\text{IncreaseSales} \sqsubseteq \exists +_ \text{infBy} . \text{StayCompetitive}$$

$$\text{StayCompetitive} \sqsubseteq (= 1 +_ \text{influences} . \text{IncreaseSales})$$

CWA axioms such as OfferCards $\sqsubseteq (= 2 \text{ refinedBy} . \text{Thing})$ complete the encoding of the graph.

“What if” Scenarios Frequently, business managers want to explore “What if?” scenarios, such as “How is the evidence for/against any particular model

element affected if our organization offers cards but does not offer international banking?”.

There are two approaches to such explorations. The first, more comfortable for domain experts, who view element models as propositions, is carried out at the class level. Thus, we would add:

$$\text{OfferCards} \sqsubseteq \text{EGoal}\langle\text{SF}\rangle \quad \text{OfferInternationBanking} \sqsubseteq \text{EGoal}\langle\text{SA}\rangle$$

and then check whether `BroadRangeOfServices` is classified as a subclass of `EGoal` \langle `SF` \rangle , . . . , `EGoal` \langle `SA` \rangle respectively. One can similarly check the classification of any other component, such as `IncreaseRevenue`, to see the effect of these assumptions on it.

One might also want to answer a different question: “Is it possible to fully satisfy `BroadRangeOfServices`?” At its simplest, this is just adding the axiom

$$\text{BroadRangeOfServices} \sqsubseteq \text{EGoal}\langle\text{SF}\rangle$$

and wait for the reasoner to detect any inconsistent concepts. Using the ability of DLs to represent incomplete information, one could of course also explore less precise scenarios (e.g., “What if we offer cards *or* international banking”).

A final variant of exploration, supported for goals in [6], is finding what (minimal) set of tasks and domain assumptions must hold if some goal is to be achieved. For this purpose one can add axioms corresponding to “predicate completion”, which end up *defining* AND and OR nodes in terms of the classes that refine them. Standard DL reasoning would then indicate what task must be executed in all circumstances if `OfferCards` is to be fully supported. However, one must use *abduction* to find a set of tasks that are sufficient to satisfy it. Unfortunately, abduction for highly expressive languages such as OWL2 has not been studied. (In [6], this is achieved using *min-SAT* algorithms for the propositional encoding.)

The alternative approach to studying scenarios, more natural to those familiar with DL modeling, would be to create an A-Box with individuals describing the particular goals, etc. being considered. For example, it would contain A-Box assertions such as `BroadRangeOfServices`(`brs1`) , `OfferCards`(`oc1`) and `refines`(`oc1`,`brs1`). One can then provide *evidence* for a scenario, such as `SFGoal`(`oc1`) , and check for consequences on individuals.

The advantage of such an approach is that it does not require changing the schema (enabling better run-time monitoring), as well as allowing the co-existence of models for multiple businesses, with potentially overlapping individuals. The disadvantage is that for a single business, one essentially duplicates the concept level axioms in the A-Box.

3.4 BIM Meta-properties

Rather than simply make BIM the union of all sorts of unrelated concepts found in other business analysis languages (e.g., Vision, Mission, Strategy (BMM), Softgoal, Hardgoal (GM), Initiative (BSc)), an ontological analysis was performed on their underlying meaning. The result is a set of six meta-properties: *duration* (long-term/short-term), *likelihood of fulfillment* (high/low), *nature of*

definition (formal/informal), *scope* (broad/narrow), *number of instances* (many/few), and *perspective* (financial/ customer/ internal/ learning and growth).

New, more specialized, BIM subconcepts can now be obtained using values for these metaproperties. For example, the BMM concept of a **Vision** is a “goal with a long duration, broad scope, low chance of fulfillment, informal definition, and few instances”. Examples of Visions from our credit card organization could include “Stay competitive” or “Have a worldwide presence”.

Not all meta-properties must take on specific values in order to express a more specialized BIM concept. Thus, **Vision** does not deal with *perspectives*. And, **Softgoals/Hardgoals** from GM can just be goals with an informal/formal definition, leaving the values of other metaproperties open.

The values of metaproperties at the class level do not constrain class instances, but only say something about the nature of instances, that they are likely or generally conform to the expressed metaproperties.

The representation of metaproperties in DLs is known to be problematic, especially in our case, where we want the metaproperties to behave so that restricting their possible values results in subclasses. However, this is exactly the behavior one would get if the metaproperties were treated as ordinary (functional) properties. So we could just add property axioms like

$$\text{number_of_instances} \sqsubseteq_r \text{Thing} \times \{ \text{few, many} \}$$

and then define classes

$$\text{Vision} \equiv \text{Goal} \sqcap (\text{number_of_instances} : \text{few}) \sqcap \dots$$

$$\sqcap (\text{nature_of_defintion} : \text{informal})$$

$$\text{SoftGoal} \equiv \text{Goal} \sqcap (\text{nature_of_defintion} : \text{informal})$$

DL reasoning would then automatically classify Vision as a subclass of SoftGoal. The main difficulty with the above approach is that this conceptual model no longer makes sense intuitively, since it associates with individual goal g , belonging to Vision, (which I might be pursuing tomorrow), the property *number_of_instances*, with value *few*. It would be much more desirable to be able to have real meta-properties of classes, but then state that, for a group of such metaproperties, value restrictions result in subclasses at the class (rather than metaclass) level.

Ideally, one would be able to state rules dealing with meta-properties, like *duration*, such as the following (*namedC/1* is a predicate that is true of atomic concept names used in DL axioms) :

$$?C \sqsubseteq ?D \quad :- \text{namedC}(?C), \text{namedC}(?D), \text{not } \text{diff_duration}(?C, ?D).$$

$$\text{diff_duration}(?C, ?D) \quad :- \text{duration}(?C, ?X), \text{duration}(?D, ?Y), ?X \neq ?Y.$$

Since *duration* is functional, this says that C is a subclass of D as long as D has not been specified to have a different meta-property value than C (i.e., D’s duration is unrestricted, or restricted to the same value as C’s). The idea is that such rules are interpreted as in Logic Programming, with negation as failure. (The precise semantics of such rules is given in Section 4.1.)

Since we want to do this for an entire set of meta-properties, we could try to use the idea of HiLog [13] to allow variables ranging over named properties, stating rules like

different_on_some_metaProp(?C,?D) : –
 namedC(?C), *namedC*(?D), *namedI*(?V), *namedI*(?W),
 ?*MP* ∈ {*duration*, *scope*, ...}, ?*MP*(?C,?V), ?*MP*(?D,?W),
 ?V ≠?W.

or simply use ternary P-assertions of the form *hasValue*(?C,?MP,?V) in the formula above.

4 More on Parametric Concepts and Rules

The Galen ontology of medical concepts [14] provides further evidence for the utility of parametric concepts/axioms. Consider the pervasive use of so-called **Selectors**. Here is one example of its use in the OWL translation of Galen (shortened by eliminating ‘Object’ vs ‘Data’, and URIs):

```
Declaration(Class(#LeftEye))
EquivalentClasses(#LeftEye
  IntersectionOf(SomeValuesFrom(#hasLeftRightSelector #leftSelection)
    #Eye))
Declaration(Class(#RightEye))
EquivalentClasses(#RightEye
  IntersectionOf(SomeValuesFrom(#hasLeftRightSelector #rightSelection)
    #Eye))
```

The following parametric declaration

```
Declaration(Class(#Eye<?LR>), ?LR in {#leftSelection,#rightSelection})
EquivalentClasses(#Eye<?LR>
  IntersectionOf(SomeValuesFrom(#hasLeftRightSelector ?LR) #Eye))
```

is meant to capture the two axioms, using an enumeration of possible concept values for the variables. Instead of the name #LeftEye, we would then use #Eye(#leftSelection), or more likely abbreviate the values, and say #Eye(#left). Both #Eye(#left) and #Eye(?V) could then be used in axioms. If this was the only example, the gain would not be much. But there are far more complex definitions involving #hasLeftRightSelector. And the above kind of repetition occurs for everything we have two in our body due to vertical symmetry. Also, there are other selectors, including #hasPositionalSelector, #hasMedialLateralSelector, #hasAnteriorPosteriorSelector, some with more than 2 values, while some concepts, such as #LeftInferiorPulmonaryVein, combine multiple selectors.

In fact a grep of the Galen OWL files revealed over 26,000 lines containing **Selector** (and naming in Galen is very systematic). So our approach would not only eliminate roughly half these axioms, but, more importantly would make *maintenance of the ontology much easier and likely to be correct*, by lessening the chance of errors when the definitions are modified later, since it is highly likely that both definitions need to be changed the same way.

Note also that in BIM, the set of qualitative values such as StrongEvidenceFor (SF), etc. might be extended to include more alternatives, such as VeryStrongEvidenceFor (VSF). For our above axiomatization of evidence propagation, this could be handled by adding only three more P-assertions: *EvidValues*(VSF), *EvidValues*(VSA), and *complement*(VSF,VSA) – clearly showing that we had captured significant patterns in our rules.

4.1 Formal and Computational Aspects

The following is a sketch of the simplest formalization of the hybrid DL+rule language we used to give the semantics of BIM. The set of primitive identifiers is split into *atomic* ones and *parametric* ones. Then axioms are formed as usual, except that parametric identifiers require atomic primitive constants or variables as arguments, and variables may occur alone as identifiers in axioms. However, any non-ground DL axiom must appear as the head of a rule, whose body binds positively all the variables in the axiom. Rules have the form

$$\gamma(\mathbf{X}) : - r_1(\mathbf{Y}_1), \dots, r_k(Y_k), \mathbf{not} s_1(\mathbf{Z}_1), \dots, \mathbf{not} s_m(Z_m)$$

where r_i and s_j are P-predicates (i.e., not in the signature of the DL), and all variables in \mathbf{X} and \mathbf{Z}_j appear among the variables in \mathbf{Y}_i for some i . The P-atoms in the body are constructed using variables or constants, some of which are *atomic* identifiers from the DL. γ is either another P-atom, or a DL axiom with free parameters \mathbf{X} . When $k = m = 0$, if γ is a P-atom then we have a P-assertion, such as *EvidValues(SF)*; otherwise, it is an ordinary (ground) DL-axiom.

The semantics of the resulting hybrid system obeys the desirable property of “modularity of reasoning” [12], by (i) using the rules first to obtain a complete set of variable-free DL axioms, and then (ii) using pure DL reasoning on the result. The semantics of rules are the standard stable-model semantics of logic programming with default negation (see [12] for a summary), with the Herbrand universe of a set of rules consisting of constants appearing in P-assertions or atomic primitive constants occurring in ground axioms. The semantics of DL are also standard, except that names of the form $C\langle d_1, \dots \rangle$ receive interpretation as atomic concepts, when there are no variables in the arguments.

In our case the rules are restricted to be non-recursive, so there are no problems with infinite Herbrand universes, decidability and complexity in part (i): one can use bottom-up evaluation as for stratified Datalog⁻; so the complexity will likely reduce to that of part (ii), since we are using a fairly expressive DL. (The precise details of this formalization can be found at <http://www.cs.rutgers.edu/~borgida/BIM/dl12.appendix.pdf>.)

As usual, the semantics should not be taken as guide to preprocess the KB and eliminate rules. First, if the benefits of parametric concepts for KB maintenance are to be realized, then the rule format should be maintained for editing, etc. Second, lightweight type-checking techniques used in Programming Languages can be used to detect certain errors in axioms without theorem proving. Third, even for absorption in tableau implementations one can perform unification to see if the parameterized axiom should be applied. Only experimental evaluation can tell whether this would result in speed-ups in an ontology like Galen, where thousands of axioms might be eliminated.

5 Summary

The presentation of BIM semantics as translation into DL provided several potentially interesting observations for the DL community.

Foremost, it led us to consider **parametric** *concepts, axioms* and *rules*. We have only scratched the surface of this area, and there remain lots of formal questions on how far one can push this in terms generalizing the syntax, semantics, complexity, and implementations.

In addition, we provided a novel way to express so-called “goal reasoning” using DL constructors. This translation makes possible the posting of goal models on the Semantic Web, and made evident the possibility of a useful “at least k subgoals need to be satisfied” variant of AND decomposition. However, in order to compete with [6], this requires more research in DL on abduction in languages more expressive than \mathcal{ALC} : the minimal language needed in our translation requires the ability to state that attributes are functional.

Acknowledgments We are very grateful to Daniel Amyot, Daniele Barone, Lei Jiang, and Eric Yu for co-developing and applying BIM.

References

1. Business motivation model, version 1.0. Object Management Group (2004).
2. Kaplan, R. S., Norton, D. P.: Strategy maps: converting intangible assets into tangible outcomes. Harvard Business School Press (2004)
3. Kaplan, R. S., Norton, D. P.: Balanced scorecard: translating strategy into action. Harvard Business School Press (1996).
4. Yu, E.: Towards modeling and reasoning support for early-phase requirements engineering. In: Proc. IEEE Int. Symp. on Req. Eng. (1997).
5. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming 20(1-2) (1993).
6. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal reasoning techniques for goal models. Journal of Data Semantics (2004).
7. Barone, D., Yu, E., Won, J., Jiang, L., Mylopoulos, J.: Enterprise modeling for business intelligence. In: Proc. PoEM'10 (2010).
8. Jiang, L., Barone, D., Amyot, D., Mylopoulos, J.: Strategic models for business intelligence: reasoning about opportunities and threats. In Proc. ER'11 (2011)
9. Barone, D., Jiang, L., Amyot, D., Mylopoulos, J.: Composite indicators for business intelligence. In Proc. ER'11 (2011).
10. Berardi, D., Calvanese, D., de Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence (2005) .
11. de Giacomo, G., Lenzerini, M., Rosati, R.: Higher-order description logics for domain metamodeling. In: Proc. AAAI-11 (2011)
12. Rosati, R.: Integrating ontologies and rules: semantic and computational issues. In: Reasoning Web'06 (2006).
13. Chen, W., Kiffer, M., Warren, D. S.: HILOG: a foundation for higher-order logic programming. J. Logic Programming 15(3) (1993).
14. OpenGALEN. <http://www.opengalen.org/>

Gödel Negation Makes Unwitnessed Consistency Crisp^{*}

Stefan Borgwardt, Felix Distel, and Rafael Peñaloza

Faculty of Computer Science
TU Dresden, Dresden, Germany
[stefborg,felix,penaloza]@tcs.inf.tu-dresden.de

Abstract. Ontology consistency has been shown to be undecidable for a wide variety of fairly inexpressive fuzzy Description Logics (DLs). In particular, for any t-norm “starting with” the Lukasiewicz t-norm, consistency of crisp ontologies (w.r.t. witnessed models) is undecidable in any fuzzy DL with conjunction, existential restrictions, and (residual) negation. In this paper we show that for any t-norm with Gödel negation, that is, any t-norm not starting with Lukasiewicz, ontology consistency for a variant of fuzzy *SHOI* is linearly reducible to crisp reasoning, and hence decidable in exponential time. Our results hold even if reasoning is not restricted to the class of witnessed models only.

1 Introduction

Fuzzy Description Logics (DLs) were introduced over a decade ago to represent and reason with vague or imprecise knowledge. Since their introduction, several variants of these logics have been studied; in fact, in addition to the constructors and kinds of axioms used, fuzzy DLs have an additional degree of liberty in the choice of the t-norm that specifies the semantics. An extensive, although slightly outdated survey of the area can be found in [18].

Very recently, it was shown that some fuzzy DLs lose the finite model property in the presence of GCIs [3]. This eventually led to a series of undecidability results [1, 2, 12, 10]. Most notably, for t-norms that “start with” the Lukasiewicz t-norm, consistency of *crisp* ontologies becomes undecidable for the inexpressive fuzzy DL $\otimes\text{-}\mathcal{MEL}$, which allows only the constructors conjunction, existential restrictions and (residual) negation [10].

So far, the only known decidability results for fuzzy DLs rely on a restriction of the expressivity: either by allowing only finitely-valued semantics [6, 8], by limiting the terminological knowledge to be acyclic or unfoldable [14, 11, 5], or by using the very simple Gödel semantics [4, 21–23]. Moreover, with very few exceptions [6, 8], reasoning is usually restricted to the class of witnessed models.¹ In fact, witnessed models were introduced in [14] to correct the previous algorithms for fuzzy DL reasoning.

^{*} Partially supported by the DFG under grant BA 1122/17-1 and in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

¹ All fuzzy logics with finitely-valued semantics have the witnessed model property.

Table 1. The three fundamental continuous t-norms

Name	t-norm ($x \otimes y$)	t-conorm ($x \oplus y$)	residuum ($x \Rightarrow y$)
Gödel	$\min\{x, y\}$	$\max\{x, y\}$	$\begin{cases} 1 & \text{if } x \leq y \\ y & \text{otherwise} \end{cases}$
product	$x \cdot y$	$x + y - x \cdot y$	$\begin{cases} 1 & \text{if } x \leq y \\ y/x & \text{otherwise} \end{cases}$
Lukasiewicz	$\max\{x + y - 1, 0\}$	$\min\{x + y, 1\}$	$\min\{1 - x + y, 1\}$

In this paper we show that for any t-norm with Gödel negation, ontology consistency is decidable in the very expressive fuzzy DL $\otimes\text{-}\mathcal{SHOI}^{-\forall}$, which is the sub-logic of $\otimes\text{-}\mathcal{SHOI}$ where value restrictions \forall are not allowed. For these logics, ontology consistency w.r.t. general models is linearly reducible to consistency of a crisp \mathcal{SHOI} ontology, and hence decidable in exponential time. In particular, this holds for the product t-norm. We emphasize that our proofs do not depend on the models being witnessed or not, hence decidability is shown for reasoning w.r.t. both, general models and witnessed models.

Since a t-norm has Gödel negation iff it does not start with the Lukasiewicz t-norm [17], this yields a full characterization of the decidability of ontology consistency (w.r.t. witnessed models) for all fuzzy DLs between $\otimes\text{-}\mathcal{NEL}$ and $\otimes\text{-}\mathcal{SHOI}^{-\forall}$. We also provide the first decidability results w.r.t. general models for infinitely-valued, non-idempotent fuzzy DLs.

2 T-norms without Zero Divisors

Mathematical fuzzy logic [13] generalizes classical logic by allowing all the real numbers from the interval $[0, 1]$ as truth values. The interpretation of the different logical constructors depends on the choice of a triangular norm (t-norm for short). A *t-norm* is an associative, commutative, and monotone binary operator on $[0, 1]$ that has 1 as its unit element. The dual operator of a t-norm \otimes is the *t-conorm* \oplus defined as $x \oplus y = 1 - ((1 - x) \otimes (1 - y))$. Notice that 0 is the unit of the t-conorm, and hence

$$x \oplus y = 0 \text{ iff } x = 0 \text{ and } y = 0. \quad (1)$$

Every *continuous* t-norm \otimes defines a unique residuum \Rightarrow such that $x \otimes y \leq z$ iff $y \leq x \Rightarrow z$ for all $x, y, z \in [0, 1]$. It is easy to see that for all $x, y \in [0, 1]$

- $x \Rightarrow y = \sup\{z \in [0, 1] \mid x \otimes z \leq y\}$,
- $1 \Rightarrow x = x$, and
- $x \leq y$ iff $x \Rightarrow y = 1$.

Based on the residuum, one can define the unary *precomplement* $\ominus x = x \Rightarrow 0$. Three important continuous t-norms are the Gödel, product and Lukasiewicz t-norms shown in Table 1, together with their t-conorms and residua. These are

fundamental in the sense that every continuous t-norm can be described as an ordinal sum of copies of these t-norms [20].

In this paper, we are interested in t-norms that do not have zero divisors. An element $x \in (0, 1)$ is called a *zero divisor* for \otimes if there is a $z \in (0, 1)$ such that $x \otimes z = 0$. Of the three fundamental continuous t-norms, only the Lukasiewicz t-norm has zero divisors. In fact, every element of the interval $(0, 1)$ is a zero divisor for this t-norm. The Gödel and product t-norms are just two elements of the uncountable class of continuous t-norms without zero divisors.

Proposition 1. *For any t-norm \otimes without zero divisors and every $x \in [0, 1]$,*

1. $x \Rightarrow y = 0$ iff $x > 0$ and $y = 0$, and
2. $\ominus x = 0$ iff $x > 0$.

Proof. For the first claim, we prove only the *if* direction, since the other direction is known to hold for every t-norm [17]. Assume that $x > 0$ and $y = 0$. Then $x \Rightarrow y = x \Rightarrow 0 = \sup\{z \mid z \otimes x = 0\}$. Since \otimes has no zero divisors, $z \otimes x > 0$ for all $z > 0$. Therefore $\{z \mid z \otimes x = 0\} = \{0\}$ and thus $x \Rightarrow y = 0$. The second statement follows from the first one since $\ominus x = x \Rightarrow 0$. \square

In particular, this implies that if the t-norm \otimes does not have zero divisors, then its precomplement is the so-called *Gödel negation*, i.e. for every $x \in [0, 1]$,

$$\ominus x = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{otherwise.} \end{cases}$$

It can be shown that the converse also holds: if the precomplement is the Gödel negation, then the t-norm has no zero divisors.

We now define the function $\mathbb{1}$ that maps fuzzy truth values to crisp truth values by setting, for all $x \in [0, 1]$,

$$\mathbb{1}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0. \end{cases}$$

For a t-norm without zero divisors it follows from Proposition 1 that $\mathbb{1}(x) = \ominus \ominus x$ for all $x \in [0, 1]$. This function is compatible with negation, the t-norm, the corresponding t-conorm, implication and suprema.

Lemma 2. *Let \otimes be a t-norm without zero divisors. For all $x, y \in [0, 1]$ and all non-empty sets $X \subseteq [0, 1]$ it holds that*

1. $\mathbb{1}(\ominus x) = \ominus \mathbb{1}(x)$,
2. $\mathbb{1}(x \otimes y) = \mathbb{1}(x) \otimes \mathbb{1}(y)$,
3. $\mathbb{1}(x \oplus y) = \mathbb{1}(x) \oplus \mathbb{1}(y)$,
4. $\mathbb{1}(x \Rightarrow y) = \mathbb{1}(x) \Rightarrow \mathbb{1}(y)$, and
5. $\mathbb{1}(\sup\{x \mid x \in X\}) = \sup\{\mathbb{1}(x) \mid x \in X\}$.

Proof. It holds that $\mathbb{1}(\ominus x) = \ominus \ominus x = \ominus \mathbb{1}(x)$ which proves 1. Since \otimes does not have zero divisors, $x \otimes y = 0$ iff $x = 0$ or $y = 0$. This shows that

$$\mathbb{1}(x \otimes y) = 0 \text{ iff } \mathbb{1}(x) \otimes \mathbb{1}(y) = 0. \quad (2)$$

Both $\mathbb{1}(x \otimes y)$ and $\mathbb{1}(x) \otimes \mathbb{1}(y)$ can only have the values 0 or 1. Hence, (2) proves the second statement. Following similar arguments we obtain from (1) that $\mathbb{1}(x \oplus y) = 0$ holds iff $\mathbb{1}(x) \oplus \mathbb{1}(y) = 0$, thus proving 3. We use Proposition 1 to prove 4:

$$\begin{aligned} \mathbb{1}(x \Rightarrow y) &= \begin{cases} 1 & \text{iff } x = 0 \text{ or } y > 0 \\ 0 & \text{iff } x > 0 \text{ and } y = 0 \end{cases} = \begin{cases} 1 & \text{iff } \mathbb{1}(x) = 0 \text{ or } \mathbb{1}(y) = 1 \\ 0 & \text{iff } \mathbb{1}(x) = 1 \text{ and } \mathbb{1}(y) = 0 \end{cases} \\ &= \mathbb{1}(x) \Rightarrow \mathbb{1}(y). \end{aligned}$$

To prove 5, observe that $\sup X = 0$ iff $X = \{0\}$. Thus,

$$\begin{aligned} \mathbb{1}(\sup X) = 0 &\Leftrightarrow \sup X = 0 \Leftrightarrow X = \{0\} \\ &\Leftrightarrow \{\mathbb{1}(x) \mid x \in X\} = \{0\} \Leftrightarrow \sup\{\mathbb{1}(x) \mid x \in X\} = 0. \end{aligned}$$

□

Notice that in general $\mathbb{1}$ is not compatible with the infimum. Consider for example the set $X = \{\frac{1}{n} \mid n \in \mathbb{N}\}$. Then $\inf X = 0$ and hence $\mathbb{1}(\inf X) = 0$, but $\inf\{\mathbb{1}(\frac{1}{n}) \mid n \in \mathbb{N}\} = \inf\{1\} = 1$.

3 The Fuzzy DL \otimes - $\mathcal{SHOI}^{-\forall}$

A fuzzy description logic usually inherits its syntax from the underlying crisp description logic. We consider the constructors of \mathcal{SHOI} with the addition of \rightarrow , which in the crisp case can be expressed by \sqcup and \neg .

Definition 3 (syntax). Let \mathbf{N}_C , \mathbf{N}_R , and \mathbf{N}_I , be disjoint sets of concept, role, and individual names, respectively, and $\mathbf{N}_R^+ \subseteq \mathbf{N}_R$ be a set of transitive role names. The set of (complex) roles is $\mathbf{N}_R \cup \{r^- \mid r \in \mathbf{N}_R\}$. \otimes - \mathcal{SHOI} (complex) concepts are constructed by the following syntax rule:

$$C ::= A \mid \top \mid \perp \mid \{a\} \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid C \rightarrow C \mid \exists s.C \mid \forall s.C,$$

where A is a concept name, a is an individual name, and s is a complex role.

The *inverse* of a complex role s (denoted by \bar{s}) is s^- if $s \in \mathbf{N}_R$ and r if $s = r^-$. A role s is *transitive* if either s or \bar{s} belongs to \mathbf{N}_R^+ .

Given a continuous t-norm \otimes , concepts in the fuzzy DL \otimes - \mathcal{SHOI} are interpreted by functions specifying the membership degree of each domain element to the concept. The interpretation of the constructors is based on the t-norm \otimes and the induced operators \oplus , \Rightarrow , and \ominus .

Definition 4 (semantics). An interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the domain $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a function that assigns to every concept name A a function $A^{\mathcal{I}}: \Delta^{\mathcal{I}} \rightarrow [0, 1]$, to every individual name a an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, and to every role name r a function $r^{\mathcal{I}}: \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0, 1]$ such that $r^{\mathcal{I}}(x, y) \otimes r^{\mathcal{I}}(y, z) \leq r^{\mathcal{I}}(x, z)$ holds for all $x, y, z \in \Delta^{\mathcal{I}}$ if $r \in \mathbf{N}_{\mathbb{R}}^+$. The function $\cdot^{\mathcal{I}}$ is extended to complex roles and concepts as follows for every $x, y \in \Delta^{\mathcal{I}}$,

- $(r^-)^{\mathcal{I}}(x, y) = r^{\mathcal{I}}(y, x)$,
- $\top^{\mathcal{I}}(x) = 1$, $\perp^{\mathcal{I}}(x) = 0$,
- $\{a\}^{\mathcal{I}}(x) = 1$ if $a^{\mathcal{I}} = x$ and 0 otherwise,
- $(\neg C)^{\mathcal{I}}(x) = \ominus C^{\mathcal{I}}(x)$,
- $(C_1 \sqcap C_2)^{\mathcal{I}}(x) = C_1^{\mathcal{I}}(x) \otimes C_2^{\mathcal{I}}(x)$, $(C_1 \sqcup C_2)^{\mathcal{I}}(x) = C_1^{\mathcal{I}}(x) \oplus C_2^{\mathcal{I}}(x)$,
- $(C_1 \rightarrow C_2)^{\mathcal{I}}(x) = C_1^{\mathcal{I}}(x) \Rightarrow C_2^{\mathcal{I}}(x)$,
- $(\exists s.C)^{\mathcal{I}}(x) = \sup_{z \in \Delta^{\mathcal{I}}} s^{\mathcal{I}}(x, z) \otimes C^{\mathcal{I}}(z)$, and
- $(\forall s.C)^{\mathcal{I}}(x) = \inf_{z \in \Delta^{\mathcal{I}}} s^{\mathcal{I}}(x, z) \Rightarrow C^{\mathcal{I}}(z)$.

\mathcal{I} is finite if its domain $\Delta^{\mathcal{I}}$ is finite, and crisp if $A^{\mathcal{I}}(x), r^{\mathcal{I}}(x, y) \in \{0, 1\}$ for all $A \in \mathbf{N}_{\mathbb{C}}$, $r \in \mathbf{N}_{\mathbb{R}}$, and $x, y \in \Delta^{\mathcal{I}}$.

Recall from the previous section that \neg is interpreted by the Gödel negation iff the t-norm \otimes does not have zero divisors. In particular, $(\neg C)^{\mathcal{I}}(x) \in \{0, 1\}$ holds for every concept C , interpretation \mathcal{I} , and $x \in \Delta^{\mathcal{I}}$, i.e. the value of $\neg C$ is always crisp.

Knowledge is encoded using axioms, which restrict the class of interpretations that are considered and specify a degree to which the restrictions should hold.

Definition 5 (axioms). A \otimes -SHOI-axiom is either an assertion of the form $\langle a:C, \ell \rangle$ or $\langle (a, b):s, \ell \rangle$, a GCI of the form $\langle C \sqsubseteq D, \ell \rangle$, or a role inclusion of the form $\langle s \sqsubseteq t, \ell \rangle$, where C and D are \otimes -SHOI-concepts, $a, b \in \mathbf{N}_{\mathbb{I}}$, s, t are complex roles, and $\ell \in (0, 1]$. An axiom is called crisp if $\ell = 1$.

An interpretation \mathcal{I} satisfies an assertion $\langle a:C, \ell \rangle$ if $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq \ell$ and an assertion $\langle (a, b):s, \ell \rangle$ if $s^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \geq \ell$. It satisfies the GCI $\langle C \sqsubseteq D, \ell \rangle$ if $C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) \geq \ell$ holds for all $x \in \Delta^{\mathcal{I}}$. It satisfies a role inclusion $\langle s \sqsubseteq t, \ell \rangle$ if $s^{\mathcal{I}}(x, y) \Rightarrow t^{\mathcal{I}}(x, y) \geq \ell$ holds for all $x, y \in \Delta^{\mathcal{I}}$.

A \otimes -SHOI-ontology $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ is defined by a finite set \mathcal{A} of assertions (ABox), a finite set \mathcal{T} of GCIs (TBox), and a finite set \mathcal{R} of role inclusions (RBox). It is crisp if every axiom in \mathcal{A} , \mathcal{T} , and \mathcal{R} is crisp. An interpretation \mathcal{I} is a model of this ontology if it satisfies all its axioms.

We consider also the logic \otimes -SHOI $^{-\forall}$, which restricts \otimes -SHOI by disallowing the constructor \forall . \otimes -SHOI $^{-\forall}$ -concepts, axioms and ontologies are defined in the obvious way. Notice that, contrary to the crisp case, value- and existential-restrictions are not dual. In fact, we will show in Section 4 that for every t-norm \otimes without zero divisors \otimes -SHOI is strictly more expressive than \otimes -SHOI $^{-\forall}$.

Several reasoning problems are of interest in the area of fuzzy DLs. Here we focus only on deciding whether a \otimes -SHOI (or \otimes -SHOI $^{-\forall}$) ontology is consistent; that is, whether it has a model. We will show that, if the t-norm

\otimes has no zero divisors, then consistency in $\otimes\text{-}\mathcal{SHOI}^{-\forall}$ is effectively the same problem as consistency in crisp \mathcal{SHOI} . Moreover, the precise values appearing in the axioms in the ontology are then irrelevant. The same is not true, however, for consistency in $\otimes\text{-}\mathcal{SHOI}$.

Recall that the semantics of the quantifiers require the computation of a supremum or infimum of the membership degrees of a possibly infinite set of elements of the domain. In the fuzzy DL community it is customary to restrict reasoning to a special kind of models, called witnessed models [14].

Definition 6 (witnessed). *An interpretation \mathcal{I} is called witnessed if for every $x \in \Delta^{\mathcal{I}}$, every role s and every concept C there are $y_1, y_2 \in \Delta^{\mathcal{I}}$ such that*

$$(\exists s.C)^{\mathcal{I}}(x) = s^{\mathcal{I}}(x, y_1) \otimes C^{\mathcal{I}}(y_1), \quad (\forall s.C)^{\mathcal{I}}(x) = s^{\mathcal{I}}(x, y_2) \Rightarrow C^{\mathcal{I}}(y_2).$$

In particular, if an interpretation \mathcal{I} is crisp or finite, then it is also witnessed. Witnessed models were introduced to simplify the reasoning tasks. In fact, although this concept was only formalized in [14], the earlier reasoning algorithms for fuzzy DLs semantics based on the Gödel t-norm (e.g. [23]) implicitly used only witnessed models. We show that consistency of $\otimes\text{-}\mathcal{SHOI}^{-\forall}$ -ontologies can be decided in exponential time, without restricting to witnessed models.

4 The Crisp Model Property

The existing undecidability results for fuzzy DLs all rely heavily on the fact that one can design ontologies that allow only models with infinitely many truth values. We shall see that for t-norms without zero divisors one cannot construct such an ontology in $\otimes\text{-}\mathcal{SHOI}^{-\forall}$. It is even true that all consistent $\otimes\text{-}\mathcal{SHOI}^{-\forall}$ -ontologies have a crisp model; that is, using at most two truth values.

Definition 7. *A fuzzy DL \mathcal{L} has the crisp model property if every consistent \mathcal{L} -ontology has a crisp model.*

For the rest of this paper we assume that \otimes is a continuous t-norm that does not have zero divisors, and hence has the properties described in Section 2. In particular, Lemma 2 allows us to construct a crisp interpretation from a fuzzy interpretation by simply applying the function $\mathbb{1}$.

Let \mathcal{I} be a fuzzy interpretation for the concept names $\mathbf{N}_{\mathcal{C}}$ and role names $\mathbf{N}_{\mathcal{R}}$. We construct the interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$, where $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}}$ and for all concept names $A \in \mathbf{N}_{\mathcal{C}}$, all role names $r \in \mathbf{N}_{\mathcal{R}}$, and all $x, y \in \Delta^{\mathcal{I}}$,

$$A^{\mathcal{J}}(x) := \mathbb{1}(A^{\mathcal{I}}(x)) \quad \text{and} \quad r^{\mathcal{J}}(x, y) := \mathbb{1}(r^{\mathcal{I}}(x, y)).$$

To show that \mathcal{J} is a valid interpretation, we first verify the transitivity condition for all $r \in \mathbf{N}_{\mathcal{R}}^+$ and all $x, y, z \in \Delta^{\mathcal{J}}$. From Lemma 2, we obtain

$$r^{\mathcal{J}}(x, y) \otimes r^{\mathcal{J}}(y, z) = \mathbb{1}(r^{\mathcal{I}}(x, y)) \otimes \mathbb{1}(r^{\mathcal{I}}(y, z)) = \mathbb{1}(r^{\mathcal{I}}(x, y) \otimes r^{\mathcal{I}}(y, z)).$$

Since \mathcal{I} satisfies the transitivity condition and $\mathbb{1}$ is monotonic, we have

$$\mathbb{1}(r^{\mathcal{I}}(x, y) \otimes r^{\mathcal{I}}(y, z)) \leq \mathbb{1}(r^{\mathcal{I}}(x, z)) = r^{\mathcal{J}}(x, z),$$

and thus $r^{\mathcal{J}}(x, y) \otimes r^{\mathcal{J}}(y, z) \leq r^{\mathcal{J}}(x, z)$.

Lemma 8. *For all complex roles s and $x, y \in \Delta^{\mathcal{I}}$, $s^{\mathcal{J}}(x, y) = \mathbb{1}(s^{\mathcal{I}}(x, y))$.*

Proof. If s is a role name, this follows directly from the definition of \mathcal{J} . If $s = r^-$ for some $r \in \mathbf{N}_{\mathbf{R}}$, then $s^{\mathcal{J}}(x, y) = r^{\mathcal{J}}(y, x) = \mathbb{1}(r^{\mathcal{I}}(y, x)) = \mathbb{1}(s^{\mathcal{I}}(x, y))$. \square

The interpretation \mathcal{J} preserves the compatibility of $\mathbb{1}$ with all the constructors of $\otimes\text{-SHOI}^{-\forall}$.

Lemma 9. *For every $\otimes\text{-SHOI}^{-\forall}$ -concept C and $x \in \Delta^{\mathcal{I}}$, $C^{\mathcal{J}}(x) = \mathbb{1}(C^{\mathcal{I}}(x))$.*

Proof. We use induction over the structure of C . The claim holds trivially for $C = \perp$ and $C = \top$. For $C = A \in \mathbf{N}_{\mathbf{C}}$ it follows immediately from the definition of \mathcal{J} . It also holds for $C = \{a\}$, $a \in \mathbf{N}_{\mathbf{I}}$, because $\{a\}^{\mathcal{I}}(x)$ can only take the values 0 or 1 for all $x \in \Delta^{\mathcal{I}}$.

Assume now that the concepts D and E satisfy $D^{\mathcal{J}}(x) = \mathbb{1}(D^{\mathcal{I}}(x))$ and $E^{\mathcal{J}}(x) = \mathbb{1}(E^{\mathcal{I}}(x))$ for all $x \in \Delta^{\mathcal{I}}$. In the case where $C = D \sqcap E$, Lemma 2 yields that for all $x \in \Delta^{\mathcal{I}}$

$$\begin{aligned} C^{\mathcal{J}}(x) &= D^{\mathcal{J}}(x) \otimes E^{\mathcal{J}}(x) = \mathbb{1}(D^{\mathcal{I}}(x)) \otimes \mathbb{1}(E^{\mathcal{I}}(x)) \\ &= \mathbb{1}(D^{\mathcal{I}}(x) \otimes E^{\mathcal{I}}(x)) = \mathbb{1}(C^{\mathcal{I}}(x)). \end{aligned}$$

Likewise, the compatibility of $\mathbb{1}$ with the t-conorm, the residuum, and the negation entails the result for the cases $C = D \sqcup E$, $C = D \rightarrow E$, and $C = \neg D$.

For $C = \exists s.D$, where s is a complex role and D is a concept description satisfying $D^{\mathcal{J}}(x) = \mathbb{1}(D^{\mathcal{I}}(x))$ for all $x \in \Delta^{\mathcal{I}}$, we obtain

$$\begin{aligned} \mathbb{1}(C^{\mathcal{I}}(x)) &= \mathbb{1}((\exists s.D)^{\mathcal{I}}(x)) = \mathbb{1}\left(\sup_{y \in \Delta^{\mathcal{I}}} \{s^{\mathcal{I}}(x, y) \otimes D^{\mathcal{I}}(y)\}\right) \\ &= \sup_{y \in \Delta^{\mathcal{I}}} \{\mathbb{1}(s^{\mathcal{I}}(x, y)) \otimes \mathbb{1}(D^{\mathcal{I}}(y))\} \end{aligned} \quad (3)$$

because $\mathbb{1}$ is compatible with the supremum and the t-norm. Lemma 8 yields

$$\sup_{y \in \Delta^{\mathcal{I}}} \{\mathbb{1}(r^{\mathcal{I}}(x, y)) \otimes \mathbb{1}(D^{\mathcal{I}}(y))\} = \sup_{y \in \Delta^{\mathcal{I}}} \{r^{\mathcal{J}}(x, y) \otimes D^{\mathcal{J}}(y)\} = (\exists r.D)^{\mathcal{J}}(x). \quad (4)$$

Equations (3) and (4) prove $\mathbb{1}(C^{\mathcal{I}}(x)) = C^{\mathcal{J}}(x)$ for $C = \exists r.D$. \square

We can use this lemma to show that the crisp interpretation \mathcal{J} satisfies all the axioms that are satisfied by \mathcal{I} .

Lemma 10. *Let $\mathcal{O} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ be a $\otimes\text{-SHOI}^{-\forall}$ -ontology. If \mathcal{I} is a model of \mathcal{O} , then \mathcal{J} is also a model of \mathcal{O} .*

Proof. We prove that \mathcal{J} satisfies all assertions, GCIs, and role inclusions from \mathcal{O} . Let $\langle a:C, \ell \rangle$, $\ell \in (0, 1]$, be a concept assertion from \mathcal{A} . Since the assertion is satisfied by \mathcal{I} , $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq \ell > 0$ holds. Lemma 9 yields $C^{\mathcal{J}}(a^{\mathcal{J}}) = 1 \geq \ell$. The same argument can be used for role assertions.

Let now $\langle C \sqsubseteq D, \ell \rangle$ be a GCI in \mathcal{T} and $x \in \Delta^{\mathcal{I}}$. Since \mathcal{I} satisfies the GCI, we get $C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) \geq \ell > 0$. By Lemmata 2 and 9, we obtain

$$C^{\mathcal{J}}(x) \Rightarrow D^{\mathcal{J}}(x) = \mathbb{1}(C^{\mathcal{I}}(x)) \Rightarrow \mathbb{1}(D^{\mathcal{I}}(x)) = \mathbb{1}(C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)) = 1 \geq \ell,$$

and thus \mathcal{J} satisfies the GCI $\langle C \sqsubseteq D, \ell \rangle$. A similar argument, using Lemma 8 instead of Lemma 9, shows that \mathcal{J} satisfies all role inclusions in \mathcal{R} . \square

The previous results show that by applying $\mathbb{1}$ to the truth degrees we obtain a crisp model \mathcal{J} from any fuzzy model \mathcal{I} of a \otimes - $\mathcal{SHOI}^{-\forall}$ -ontology \mathcal{O} .

Theorem 11. *If \otimes is a t-norm without zero divisors, then \otimes - $\mathcal{SHOI}^{-\forall}$ has the crisp model property.*

A trivial consequence of this theorem is that every consistent \otimes - $\mathcal{SHOI}^{-\forall}$ -ontology has also a witnessed model, since every crisp model is also crisp.

Corollary 12. *If \otimes is a t-norm without zero divisors, then \otimes - $\mathcal{SHOI}^{-\forall}$ has the witnessed model property.*

In the next section we will use this result from Theorem 11 to show that \otimes - $\mathcal{SHOI}^{-\forall}$ ontology consistency can be decided in exponential time, by testing consistency of a (crisp) \mathcal{SHOI} ontology. But first, we show that value restrictions destroy the crisp model property, even if only crisp axioms are used.

Example 13. Consider the \otimes - \mathcal{SHOI} -ontology

$$\mathcal{O} = \{ \langle \top \sqsubseteq \neg\neg A, 1 \rangle, \langle a: \neg\forall r.A, 1 \rangle \}.$$

The interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with $\Delta^{\mathcal{I}} = \mathbb{N}$ (the set of all natural numbers), $a^{\mathcal{I}} = 1$, $A^{\mathcal{I}}(n) = 1/(n+1)$, $r^{\mathcal{I}}(1, n) = 1$, and $r^{\mathcal{I}}(n', n) = 0$ for all $n, n' \in \mathbb{N}$ with $n' > 1$ is a model of \mathcal{O} , and hence \mathcal{O} is consistent.

Let now \mathcal{J} be a crisp interpretation satisfying the first axiom in \mathcal{O} . Then, $A^{\mathcal{J}}(x) = 1$ for all $x \in \Delta^{\mathcal{J}}$. This implies that

$$\begin{aligned} (\forall r.A)^{\mathcal{J}}(a^{\mathcal{J}}) &= \inf_{y \in \Delta^{\mathcal{J}}} r^{\mathcal{J}}(a^{\mathcal{J}}, y) \Rightarrow A^{\mathcal{J}}(y) \\ &= \inf_{y \in \Delta^{\mathcal{J}}} r^{\mathcal{J}}(a^{\mathcal{J}}, y) \Rightarrow 1 \\ &= 1. \end{aligned}$$

And thus, $(\neg\forall r.A)^{\mathcal{J}}(a^{\mathcal{J}}) = 0$, violating the second axiom. This means that \mathcal{O} has no crisp model.

The example shows that no fuzzy DL with the constructor \forall and Gödel negation² has the crisp model property. A similar example in [14] demonstrates that no fuzzy DL with the constructors \exists and \forall and Gödel negation has the witnessed model property.

Theorem 14. *For any continuous t-norm \otimes and any fuzzy DL \otimes - \mathcal{L} having the constructors \top , \neg , and \forall , \otimes - \mathcal{L} does not have the crisp model property.*

In particular, this means that \otimes - \mathcal{SHOI} does not have the crisp model property and is strictly more expressive than \otimes - $\mathcal{SHOI}^{-\forall}$.

Corollary 15. *If \otimes is a t-norm without zero divisors, then \otimes - \mathcal{SHOI} is strictly more expressive than \otimes - $\mathcal{SHOI}^{-\forall}$.*

5 Deciding Consistency

For a given \otimes - $\mathcal{SHOI}^{-\forall}$ -ontology \mathcal{O} , we define $\text{crisp}(\mathcal{O})$ to be the crisp \mathcal{SHOI} -ontology that is obtained from \mathcal{O} by replacing all the truth values appearing in the axioms by 1. For example, for the ontology

$$\mathcal{O} = \{\langle a:C, 0.2 \rangle, \langle (a,b):r, 0.8 \rangle, \langle C \sqsubseteq D, 0.5 \rangle, \langle r \sqsubseteq s, 0.1 \rangle\}$$

we obtain

$$\text{crisp}(\mathcal{O}) = \{\langle a:C, 1 \rangle, \langle (a,b):r, 1 \rangle, \langle C \sqsubseteq D, 1 \rangle, \langle r \sqsubseteq s, 1 \rangle\}.$$

Lemma 16. *Let \mathcal{O} be a \otimes - $\mathcal{SHOI}^{-\forall}$ -ontology and \mathcal{I} be a crisp interpretation. Then \mathcal{I} is a model of \mathcal{O} iff it is a model of $\text{crisp}(\mathcal{O})$.*

Proof. Assume that $\text{crisp}(\mathcal{O})$ has a model \mathcal{I} . Let $\langle C \sqsubseteq D, \ell \rangle$, $\ell > 0$, be an axiom from \mathcal{O} . Since \mathcal{I} is a model of $\text{crisp}(\mathcal{O})$, it must satisfy $\langle C \sqsubseteq D, 1 \rangle$; that is, $C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) \geq 1 \geq \ell$ holds for all $x \in \Delta^{\mathcal{I}}$. Thus \mathcal{I} satisfies $\langle C \sqsubseteq D, \ell \rangle$. The proof that \mathcal{I} satisfies assertions and role inclusions is analogous. Hence \mathcal{I} is also a model of \mathcal{O} .

For the other direction, assume that \mathcal{I} satisfies $\langle C \sqsubseteq D, \ell \rangle$ with $\ell > 0$. As \mathcal{I} is a crisp interpretation it holds that $C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) \in \{0, 1\}$ for all $x \in \Delta^{\mathcal{I}}$. Together with $C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) \geq \ell > 0$ we obtain $C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) = 1$. Thus, \mathcal{I} satisfies the GCI $\langle C \sqsubseteq D, 1 \rangle$. The same argument can be used for role inclusions and assertions. Thus, \mathcal{I} is also a model of $\text{crisp}(\mathcal{O})$. \square

In particular, a \otimes - $\mathcal{SHOI}^{-\forall}$ -ontology \mathcal{O} has a crisp model iff $\text{crisp}(\mathcal{O})$ has a crisp model. Together with Theorem 11, this shows that a \otimes - $\mathcal{SHOI}^{-\forall}$ -ontology \mathcal{O} is consistent iff $\text{crisp}(\mathcal{O})$ has a crisp model. Therefore, one can use any reasoning procedure for crisp \mathcal{SHOI} to decide consistency of \otimes - $\mathcal{SHOI}^{-\forall}$ -ontologies. Reasoning in crisp \mathcal{SHOI} is known to be EXPTIME-complete [15]. Recall that under crisp semantics, value restrictions can be expressed by negation and existential restrictions, and hence, crisp \mathcal{SHOI} is equivalent to crisp $\mathcal{SHOI}^{-\forall}$.

² Recall that a fuzzy DL has Gödel negation iff its semantics is based on a t-norm without zero divisors.

Corollary 17. *Deciding consistency in \otimes - $\mathcal{SHOI}^{-\forall}$ is EXPTIME-complete.*

Lemma 16 and Theorem 11 still hold when we restrict the semantics to the slightly less expressive logics \otimes - $\mathcal{SHO}^{-\forall}$, which does not allow for inverse roles, or \otimes - $\mathcal{SI}^{-\forall}$ which does not allow for nominals and role hierarchies. The crisp DLs \mathcal{SHO} and \mathcal{SI} are known to have the finite model property [16, 19], and \otimes - $\mathcal{SI}^{-\forall}$ and \otimes - $\mathcal{SHO}^{-\forall}$ inherit the finite model property from their crisp counterparts.

Theorem 18. *The logics \otimes - $\mathcal{SHO}^{-\forall}$ and \otimes - $\mathcal{SI}^{-\forall}$ and their sublogics have the finite model property.*

6 Conclusions

In this paper we have described a family of expressive fuzzy DLs for which ontology consistency is decidable. More precisely, we have shown that if \otimes is a t-norm without zero divisors, consistency of \otimes - $\mathcal{SHOI}^{-\forall}$ ontologies is EXPTIME-complete, and hence as hard as consistency of (crisp) \mathcal{SHOI} ontologies. Our construction shows that the fuzzy values appearing in \otimes - $\mathcal{SHOI}^{-\forall}$ ontologies are irrelevant for consistency: a \otimes - $\mathcal{SHOI}^{-\forall}$ ontology \mathcal{O} has a (fuzzy) model iff its crisp variant $\text{crisp}(\mathcal{O})$, where the degrees of all the axioms in \mathcal{O} are changed to 1, has a crisp model. This implies that \otimes - $\mathcal{SHOI}^{-\forall}$ has the crisp model property, and hence also the witnessed model property. If the constructor \forall is also allowed, hence obtaining the logic \otimes - \mathcal{SHOI} , then these properties do not hold anymore.

For other reasoning problems such as entailment and subsumption it is unknown whether they are decidable in \otimes - $\mathcal{SHOI}^{-\forall}$. In [7] it is shown for \otimes - \mathcal{SHOI} with witnessed models that subsumption and entailment, as well as computing the best subsumption and entailment degrees, cannot be reduced to crisp reasoning by simply mapping all nonzero truth degrees to 1. We conjecture that this is also the case in \otimes - $\mathcal{SHOI}^{-\forall}$.

It has recently been shown that if the t-norm \otimes has zero divisors, then consistency of crisp ontologies in the very inexpressive fuzzy DL \otimes - \mathcal{NEL} w.r.t. witnessed models [10] and w.r.t. general models [9] is undecidable.³ Combining these results, we obtain a characterization of the decidability of consistency w.r.t. witnessed and general models for all fuzzy DLs between \otimes - \mathcal{NEL} and \otimes - $\mathcal{SHOI}^{-\forall}$: it is decidable (in EXPTIME) iff \otimes has no zero divisors.

In future work, we plan to study reasoning problems in fuzzy DLs allowing for value restrictions without the restriction to witnessed models. In this direction it is worth looking at the decidability results for \otimes - \mathcal{ALC} with product t-norm w.r.t. *quasi-witnessed* models [11].

References

1. Baader, F., Peñaloza, R.: Are fuzzy description logics with general concept inclusion axioms decidable? In: Proc. of the 2011 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'11). pp. 1735–1742. IEEE Press (2011)

³ \otimes - \mathcal{NEL} is the sublogic of \otimes - $\mathcal{SHOI}^{-\forall}$ that allows only the constructors \top , \neg and \exists .

2. Baader, F., Peñaloza, R.: On the undecidability of fuzzy description logics with GCIs and product t-norm. In: Proc. of 8th Int. Symp. Frontiers of Combining Systems (FroCoS'11). pp. 55–70. Springer-Verlag (2011)
3. Bobillo, F., Bou, F., Straccia, U.: On the failure of the finite model property in some fuzzy description logics. Fuzzy Sets and Systems 172(23), 1–12 (2011)
4. Bobillo, F., Delgado, M., Gómez-Romero, J., Straccia, U.: Fuzzy description logics under Gödel semantics. Int. J. of Approx. Reasoning 50(3), 494–514 (2009)
5. Bobillo, F., Straccia, U.: Fuzzy description logics with general t-norms and datatypes. Fuzzy Sets and Systems 160(23), 3382–3402 (2009)
6. Bobillo, F., Straccia, U.: Reasoning with the finitely many-valued Łukasiewicz fuzzy description logic *SRQIQ*. Information Sciences 181, 758–778 (2011)
7. Borgwardt, S., Distel, F., Peñaloza, R.: How fuzzy is my fuzzy description logic? In: Proc. of the 6th Int. Joint Conf. on Autom. Reas. (IJCAR'12) (2012), to appear
8. Borgwardt, S., Peñaloza, R.: Description logics over lattices with multi-valued ontologies. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 768–773. AAAI Press (2011)
9. Borgwardt, S., Peñaloza, R.: Non-Gödel negation makes unwitnessed consistency undecidable. In: Proc. of the 25th Int. Workshop on Description Logics (DL 2012). CEUR Workshop Proceedings, Rome, Italy (2012), to appear
10. Borgwardt, S., Peñaloza, R.: Undecidability of fuzzy description logics. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012). AAAI Press, Rome, Italy (2012), to appear
11. Cerami, M., Esteva, F., Bou, F.: Decidability of a description logic over infinite-valued product logic. In: Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2010). AAAI Press (2010)
12. Cerami, M., Straccia, U.: On the undecidability of fuzzy description logics with GCIs with Łukasiewicz t-norm. Tech. rep., Computing Research Repository (2011), [arXiv:1107.4212v3 \[cs.LG\]](https://arxiv.org/abs/1107.4212v3). An extended version of this paper has been submitted to a journal.
13. Hájek, P.: Metamathematics of Fuzzy Logic (Trends in Logic). Springer-Verlag (2001)
14. Hájek, P.: Making fuzzy description logic more general. Fuzzy Sets and Systems 154(1), 1–15 (2005)
15. Hladik, J.: To and Fro Between Tableaus and Automata for Description Logics. Ph.D. thesis, Dresden University of Technology, Germany (2007)
16. Horrocks, I., Sattler, U., Tobies, S.: A PSpace-algorithm for deciding \mathcal{ALCN}_{R^+} -satisfiability. LTCS-Report 98-08, RWTH Aachen, Germany (1998)
17. Klement, E.P., Mesiar, R., Pap, E.: Triangular Norms. Springer-Verlag (2000)
18. Łukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics 6(4), 291–308 (2008)
19. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. Journal of Artificial Intelligence Research 23, 667–726 (2004)
20. Mostert, P.S., Shields, A.L.: On the structure of semigroups on a compact manifold with boundary. Annals of Mathematics 65, 117–143 (1957)
21. Stoilos, G., Stamou, G.B., Pan, J.Z., Tzouvaras, V., Horrocks, I.: Reasoning with very expressive fuzzy description logics. JAIR 30, 273–320 (2007)
22. Straccia, U.: Reasoning within fuzzy description logics. Journal of Artificial Intelligence Research 14, 137–166 (2001)
23. Tresp, C.B., Molitor, R.: A description logic for vague knowledge. In: Proc. of the 13th Eur. Conf. on Artificial Intelligence (ECAI'98). pp. 361–365. J. Wiley and Sons, Brighton, UK (1998)

Towards More Effective Tableaux Reasoning for CKR

Loris Bozzato¹, Martin Homola², and Luciano Serafini¹

¹ Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy

² FMFI, Comenius University, Mlynská dolina, 84248 Bratislava, Slovakia

{bozzato,serafini}@fbk.eu, homola@fmph.uniba.sk

1 Introduction

Representation of context dependent knowledge in the Semantic Web is a recently emergent issue. A number of logical formalisms with this aim have been proposed [3, 11, 14]. Among them is the DL based Contextualized Knowledge Repository (CKR) [13]. One of the mostly advocated advantages of context based knowledge representation is that reasoning procedures can be constructed by composing local reasoners running inside each context, with the obvious divide-and-conquer advantage.

We recently proposed a tableaux decision algorithm [5, 9] for the case of CKR framework based on \mathcal{ALC} DL. The algorithm extends the well known \mathcal{ALC} tableaux algorithm [6, 12] and it is based on a combination of local reasoning inside each context with a set of novel rules that propagate knowledge across the neighboring contexts. To our best knowledge, it is the only direct tableaux reasoning algorithm for contextualized DL knowledge to date: by direct we mean not based on some reduction to a single DL knowledge base, which neglects the divide-and-conquer advantage.

In this paper, we review this algorithm and we describe our initial ideas on possible optimization, including dimensional coverage caching and parallelization. In order to maximize the divide-and-conquer advantage, it is important to propagate only those symbols between local tableaux which are really needed to assure completeness. We propose a (correctness preserving) modification of three propagation rules that decreases the amount of propagation and also of related non-deterministic branching. Proofs of all theorems can be found in the accompanying technical report [9].

2 Contextualized Knowledge Repositories

We briefly introduce the basic definition of CKR, for all details see [13]. A *meta vocabulary* Γ is used to state information about contexts. It contains contextual attributes (called dimensions), their possible values and coverage relations between these values. Formally, it is a DL vocabulary that contains: (a) a finite set of individuals called *context identifiers*; (b) a finite set of roles \mathbf{A} called *dimensions*; (c) for every dimension $A \in \mathbf{A}$, a finite set of individuals D_A , called *dimensional values*, and a role \prec_A , called *coverage relation*. The number of dimensions $k = |\mathbf{A}|$ is assumed to be a fixed constant.

Dimensional vectors are used to identify each context with a specific set of dimensional values. Given a meta-vocabulary Γ with dimensions $\mathbf{A} = \{A_1, \dots, A_k\}$,

a *dimensional vector* $\mathbf{d} = \{A_{i_1}:=d_1, \dots, A_{i_m}:=d_m\}$ is a (possibly empty) set of assignments such that for every j, h , with $1 \leq j \leq h \leq m$, $d_j \in D_{A_{i_j}}$, and $j \neq h$ implies $i_j \neq i_h$. A dimensional vector \mathbf{d} is full if it assigns values to all dimensions (i.e., $|\mathbf{d}| = k$), otherwise it is partial. If it is apparent which value belongs to which dimension, we simply write $\{d_1, \dots, d_m\}$. By d_A (e_A , etc.) we denote the actual value that \mathbf{d} (\mathbf{e} , etc.) assigns to the dimension A . The *dimensional space* \mathcal{D}_Γ of Γ is the set of all full dimensional vectors of Γ .

An *object-vocabulary*, encodes knowledge inside contexts: it is a standard DL vocabulary Σ (with disjoint sets N_C of atomic concepts, N_R of roles and N_I of individuals) closed w.r.t. *concept/role qualification*. That is, for every concept or role symbol X of Σ and every (possibly partial) dimensional vector \mathbf{d} , a new symbol $X_{\mathbf{d}}$ is added to Σ , called the *qualification* of X w.r.t. \mathbf{d} . If \mathbf{d} is partial then $X_{\mathbf{d}}$ is partially qualified, if \mathbf{d} is full, it is fully qualified. Qualified symbols are used inside contexts to refer to the meaning of symbols w.r.t. some other context. This will become apparent from the semantics. Contexts and CKR knowledge bases are formally defined as follows.

Definition 1 (Context). Given a pair of meta and object vocabularies $\langle \Gamma, \Sigma \rangle$, a context is a triple $\langle \mathcal{C}, \dim(\mathcal{C}), K(\mathcal{C}) \rangle$ where: \mathcal{C} is a context identifier of Γ ; $\dim(\mathcal{C})$ is a full dimensional vector of \mathcal{D}_Γ ; and $K(\mathcal{C})$ is an *ALC* knowledge base over Σ .

Definition 2 (Contextualized Knowledge Repository). Given a pair of meta and object vocabularies $\langle \Gamma, \Sigma \rangle$, a CKR knowledge base (CKR) is a pair $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{C} \rangle$ where \mathfrak{C} is a set of contexts on $\langle \Gamma, \Sigma \rangle$ and \mathfrak{M} , called meta knowledge, is a DL knowledge base over Γ such that:

- (a) for $A \in \mathbf{A}$ and $d, d' \in D_A$, if $\mathfrak{M} \models A(\mathcal{C}, d)$ and $\mathfrak{M} \models A(\mathcal{C}, d')$ then $\mathfrak{M} \models d = d'$;
- (b) for $\mathcal{C} \in \mathfrak{C}$ with $\dim(\mathcal{C}) = \mathbf{d}$ and for $A \in \mathbf{A}$, we have $\mathfrak{M} \models A(\mathcal{C}, d_A)$;
- (c) the relation $\{\langle d, d' \rangle \mid \mathfrak{M} \models d \prec_A d'\}$ is a strict partial order on D_A .

In the rest of the paper we assume that CKR knowledge bases are defined over some suitable vocabulary $\langle \Gamma, \Sigma \rangle$, and all concepts are in negation normal form (NNF, see [1]). We also assume the unique name assumption (UNA) for the meta knowledge (i.e., if $a \neq b$ are two different symbols then $\mathfrak{M} \not\models a = b$). This is just to avoid the confusing possibility of two contexts located as the same place in the dimensional space.

For a CKR \mathfrak{K} , we will denote by $\mathcal{C}_{\mathbf{d}}$ a context with $\dim(\mathcal{C}) = \mathbf{d}$. For $\mathbf{d}, \mathbf{e} \in \mathcal{D}_\Gamma$ and $\mathbf{B}, \mathbf{C} \subseteq \mathbf{A}$, $\mathbf{d}_{\mathbf{B}} := \{(A:=d) \in \mathbf{d} \mid A \in \mathbf{B}\}$ is the projection of \mathbf{d} w.r.t. \mathbf{B} ; and $\mathbf{d}_{\mathbf{B}} + \mathbf{e}_{\mathbf{C}} := \mathbf{d}_{\mathbf{B}} \cup \{(A:=d) \in \mathbf{e}_{\mathbf{C}} \mid A \notin \mathbf{B}\}$ is the completion of $\mathbf{d}_{\mathbf{B}}$ w.r.t. $\mathbf{e}_{\mathbf{C}}$.

An important notion is the strict (\prec) and non-strict (\preceq) coverage between dimensional values: for $d, d' \in D_A$, $d \prec d'$ if $\mathfrak{M} \models d \prec_A d'$; and $d \preceq d'$ if either $d \prec d'$ or $\mathfrak{M} \models d = d'$. Similarly, coverage for dimensional vectors: $\mathbf{d} \preceq_{\mathbf{B}} \mathbf{e}$ for some $\mathbf{B} \subseteq \mathbf{A}$ if $d_B \preceq e_B$ for each $B \in \mathbf{B}$; and $\mathbf{d} \prec_{\mathbf{B}} \mathbf{e}$ if $\mathbf{d} \preceq_{\mathbf{B}} \mathbf{e}$ and $d_B \prec e_B$ for at least one $B \in \mathbf{B}$. Also, $\mathbf{d} \preceq \mathbf{e}$ if $\mathbf{d} \preceq_{\mathbf{A}} \mathbf{e}$, and $\mathbf{d} \prec \mathbf{e}$ if $\mathbf{d} \prec_{\mathbf{A}} \mathbf{e}$. Finally coverage for contexts: $\mathcal{C}_{\mathbf{d}} \preceq \mathcal{C}_{\mathbf{e}}$ if $\mathbf{d} \preceq \mathbf{e}$, and $\mathcal{C}_{\mathbf{d}} \prec \mathcal{C}_{\mathbf{e}}$ if $\mathbf{d} \prec \mathbf{e}$. Intuitively, if $\mathcal{C}_{\mathbf{d}} \prec \mathcal{C}_{\mathbf{e}}$, then $\mathcal{C}_{\mathbf{d}}$ is the narrower and $\mathcal{C}_{\mathbf{e}}$ is the broader context.

An example CKR \mathfrak{K}_{fb} shown in Fig. 1 uses three dimensions time, location, and topic. It has four contexts associated with dimensional vectors \mathbf{sp} (general context of sports in 2010), \mathbf{fb} (football in 2010), $\mathbf{wc10}$ (FIFA World Cup 2010), and $\mathbf{nf10}$

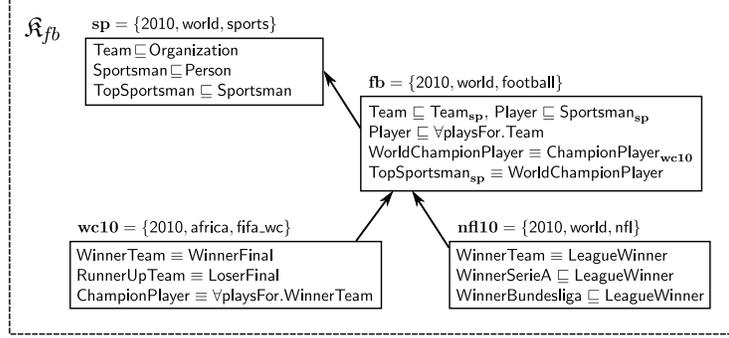


Fig. 1. Example CKR knowledge base \mathcal{R}_{fb}

(national football leagues in 2010). Axioms are placed inside each context while the associated vector is placed above it. Coverage relation \prec is visualized with arrows.

Note that in CKR built on top of more expressive logics, conditions 2 (a,c) of Definition 2 can be assured directly in the meta knowledge with respective axioms: each $A \in \mathbf{A}$ is declared functional, and each \prec_A is declared irreflexive and transitive. In \mathcal{ALC} we do not have this option, however this is not a problem, because the number of all dimensions is assumed to be finite as it is the number of contexts in a CKR. Hence after the meta knowledge is modeled, these conditions can be verified even without a reasoner (e.g., by some script). These conditions are needed to assure reasonable properties of contextual space, i.e., acyclicity, dimensional values uniquely determined [13].

CKR uses DL semantics inside each context combined with some additional semantic restrictions to ensure proper meaning of qualified symbols. A *partial DL interpretation* of a DL vocabulary Σ is a DL interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ that allows two exceptions: $\Delta^{\mathcal{I}}$ is possibly an empty set, and $\cdot^{\mathcal{I}}$ is totally defined on N_C and N_R and it is partially defined on N_I (i.e., $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ can be undefined for some $a \in N_I$). Partial interpretations need not necessarily provide denotations for all individuals of Σ . This is needed for technical reasons: intuitively, all contexts rely on the same object vocabulary Σ , but some element of Σ may not be meaningful in all contexts. Also, interpretations with empty domains are useful to treat inconsistency among contexts [13].

Definition 3 (CKR-Model). A model of a CKR \mathcal{R} is a collection $\mathcal{J} = \{\mathcal{I}_{\mathbf{d}}\}_{\mathbf{d} \in \mathcal{D}_{\mathcal{R}}}$ of partial DL interpretations (local interpretations) s.t. for all $\mathbf{d}, \mathbf{e}, \mathbf{f} \in \mathcal{D}_{\mathcal{R}}$, $\mathbf{B} \subseteq \mathbf{A}$, $A \in N_C$, $R \in N_R$, $X \in N_C \cup N_R$, $a \in N_I$:

1. $(\top_{\mathbf{d}})^{\mathcal{I}_{\mathbf{d}}} \subseteq (\top_{\mathbf{e}})^{\mathcal{I}_{\mathbf{e}}}$ if $\mathbf{d} \prec \mathbf{e}$
2. $(A_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}} \subseteq (\top_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}}$
3. $(R_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}} \subseteq (\top_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}} \times (\top_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}}$
4. $a^{\mathcal{I}_{\mathbf{e}}} = a^{\mathcal{I}_{\mathbf{d}}}$ if $\mathbf{d} \prec \mathbf{e}$ and
 - $a^{\mathcal{I}_{\mathbf{d}}}$ is defined or,
 - $a^{\mathcal{I}_{\mathbf{e}}}$ is defined and $a^{\mathcal{I}_{\mathbf{e}}} \in \Delta_{\mathbf{d}}$
5. $(X_{\mathbf{d}_{\mathbf{B}}})^{\mathcal{I}_{\mathbf{e}}} = (X_{\mathbf{d}_{\mathbf{B}+\mathbf{e}}})^{\mathcal{I}_{\mathbf{e}}}$
6. $(X_{\mathbf{d}})^{\mathcal{I}_{\mathbf{e}}} = (X_{\mathbf{d}})^{\mathcal{I}_{\mathbf{d}}}$ if $\mathbf{d} \prec \mathbf{e}$
7. $(A_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}} = (A_{\mathbf{f}})^{\mathcal{I}_{\mathbf{e}}} \cap \Delta_{\mathbf{d}}$ if $\mathbf{d} \prec \mathbf{e}$
8. $(R_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}} = (R_{\mathbf{f}})^{\mathcal{I}_{\mathbf{e}}} \cap (\Delta_{\mathbf{d}} \times \Delta_{\mathbf{d}})$ if $\mathbf{d} \prec \mathbf{e}$
9. $\mathcal{I}_{\mathbf{d}} \models \mathbf{K}(\mathcal{C}_{\mathbf{d}})$

The semantics takes care that local domains respect the coverage hierarchy (condition 1). Note that $\top_{\mathbf{d}}$ represents the domain of $\mathcal{I}_{\mathbf{d}}$ in the context where it appears. It

gives rigid meaning to individuals, however, the meaning of an individual in a super-context is independent if its meaning in a sub-context is undefined (condition 4). The interpretation of any X_f in any context \mathcal{C}_d is roofed under $(\top_f)^{\mathcal{I}_d}$ (conditions 2, 3). The meaning of X_f in some context \mathcal{C}_e is based on its context of origin \mathcal{C}_f if this context is less specific than \mathcal{C}_e (condition 6); otherwise, at least, any X_f in \mathcal{C}_d and \mathcal{C}_e must be equal on the shared part of their domains (conditions 7 and 8). Finally, each \mathcal{I}_d is a DL-model of \mathcal{C}_d (condition 9). Albeit useful for modeling, partially qualified symbols are a kind of syntactic sugar in this framework as the completed version of the symbol can always be used instead (condition 5, cf. [13]). To simplify the algorithm, we assume w.l.o.g. that the CKR on the input is always fully qualified. In the examples we use non-qualified symbols only for improving readability.

Given a CKR \mathfrak{K} and $\mathbf{d} \in \mathfrak{D}_\Gamma$, a concept C is **d-satisfiable** w.r.t. \mathfrak{K} if there exists a CKR model $\mathcal{J} = \{\mathcal{I}_e\}_{e \in \mathfrak{D}_\Gamma}$ of \mathfrak{K} such that $C^{\mathcal{I}_d} \neq \emptyset$; \mathfrak{K} is **d-satisfiable** if it has a CKR model $\mathcal{J} = \{\mathcal{I}_e\}_{e \in \mathfrak{D}_\Gamma}$ such that $\Delta_d \neq \emptyset$; \mathfrak{K} is **globally satisfiable** if it has a CKR model $\mathcal{J} = \{\mathcal{I}_e\}_{e \in \mathfrak{D}_\Gamma}$ such that $\Delta_e \neq \emptyset$ for every $e \in \mathfrak{D}_\Gamma$. An axiom α is **d-entailed** by \mathfrak{K} (denoted $\mathfrak{K} \models \mathbf{d} : \alpha$) if for every model $\mathcal{J} = \{\mathcal{I}_e\}_{e \in \mathfrak{D}_\Gamma}$ of \mathfrak{K} it holds $\mathcal{I}_d \models \alpha$. As usual, **d-entailment** can be reduced to **d-satisfiability**: in particular $\mathfrak{K} \models \mathbf{d} : C \sqsubseteq D$ iff $C \sqcap \neg D$ is not **d-satisfiable** w.r.t. \mathfrak{K} .

3 Tableaux Algorithm for CKR

We denote by $\text{clos}(C)$ the set of all syntactically correct atomic and complex concepts that occur in a concept C . The closure of a concept C w.r.t. a CKR \mathfrak{K} is $\text{clos}_{\mathfrak{K}}(C) = \text{clos}(C) \cup \{\text{clos}(\neg D \sqcup E) \mid D \sqsubseteq E \in \text{K}(\mathcal{C}) \text{ for some context } \mathcal{C} \text{ of } \mathfrak{K}\} \cup \{\text{clos}(D) \mid D(a) \in \text{K}(\mathcal{C}) \text{ for some context } \mathcal{C} \text{ of } \mathfrak{K}\} \cup \{\text{clos}(\neg \top_e \sqcup \top_f) \mid e \prec f\}$. We denote with $\mathcal{R}_{\mathfrak{K}, C}$ the set of roles $R \in N_R$ appearing in C or some $\text{K}(\mathcal{C})$ of \mathfrak{K} . The sets $\text{clos}_{\mathfrak{K}}(C)$ and $\mathcal{R}_{\mathfrak{K}, C}$ contain all possible concepts and roles relevant in order to verify **d-satisfiability** of C w.r.t. \mathfrak{K} .

The tableaux algorithm $C_{\mathcal{T}}$ for CKR decides the **d-satisfiability** of a concept C w.r.t. a CKR \mathfrak{K} : it is partly based on the well known \mathcal{ALC} tableaux algorithm [12, 6] which is extended in order to deal with multiple contexts. The algorithm works on a *completion tree*, a partial representation of a CKR model that the algorithm incrementally builds.

Definition 4 (Completion tree). *Given a CKR \mathfrak{K} , a completion tree is a triple $T = \langle V, E, \mathcal{L} \rangle$ s.t.:*

1. $\langle V, E \rangle$ is a tree, where V is an ordered set of elements with order $<_V$;
2. there is a collection $\{V_d\}_{d \in \mathfrak{D}_\Gamma}$ of sets such that $V_d \subseteq V$;
3. $E_d = \{\langle x, y \rangle \in E \mid x, y \in V_d\}$, for each $d \in \mathfrak{D}_\Gamma$;
4. $\mathcal{L} = \{\mathcal{L}_d\}_{d \in \mathfrak{D}_\Gamma}$ is a collection of labeling functions such that for each $d \in \mathfrak{D}_\Gamma$:
 - (a) $\mathcal{L}_d(x) \subseteq \text{clos}_{\mathfrak{K}}(C)$, for each $x \in V_d$;
 - (b) $\mathcal{L}_d(\langle x, y \rangle) \subseteq \mathcal{R}_{\mathfrak{K}, C}$, for each $\langle x, y \rangle \in E_d$.

In order to verify **d-satisfiability** of a concept C w.r.t. a CKR \mathfrak{K} , the algorithm initializes and then iteratively expands the tree using a number of tableaux expansion rules. To avoid infinite looping, a blocking policy adapted from Buchheit et al. [6] is used. We assume that the algorithm always adds nodes into the completion tree respecting the order $<_V$ (i.e., whenever a new node x is added, $y <_V x$ holds for all y already in V).

Table 1. CKR completion rules

\sqcap -rule: if $x \in V_d, C_1 \sqcap C_2 \in \mathcal{L}_d(x),$ $\{C_1, C_2\} \not\subseteq \mathcal{L}_d(x)$ then $\mathcal{L}_d(x) := \mathcal{L}_d(x) \cup \{C_1, C_2\}$	$\Delta\downarrow$ -rule: if $x \in V_e, \mathbf{d} \prec \mathbf{e}, \top_d \in \mathcal{L}_e(x), x \notin V_d$ then $V_d = V_d \cup \{x\}$
\sqcup -rule: if $x \in V_d, C_1 \sqcup C_2 \in \mathcal{L}_d(x),$ $\{C_1, C_2\} \cap \mathcal{L}_d(x) = \emptyset$ then $\mathcal{L}_d(x) := \mathcal{L}_d(x) \cup \{C_1\}$ or $\mathcal{L}_d(x) := \mathcal{L}_d(x) \cup \{C_2\}$	A-rule: if $x \in V_d \cap V_e, \mathbf{d} \prec \mathbf{e}$ or $\mathbf{d} \succ \mathbf{e},$ $A_f \in \mathcal{L}_d(x), A_f \notin \mathcal{L}_e(x)$ then $\mathcal{L}_e(x) := \mathcal{L}_e(x) \cup \{A_f\}$
\exists -rule: if $x \in V_d, \exists R.C \in \mathcal{L}_d(x),$ and there is no R-successor $y \in V_d$ of x s.t. $C \in \mathcal{L}_d(y)$ then $V_d := V_d \cup \{z\}$ with z new, $E_d := E_d \cup \{(x, z)\}$ $\mathcal{L}_d(\langle x, z \rangle) := \{R\}, \mathcal{L}_d(z) := \{C\}$	R-rule: if $x, y \in V_d \cap V_e, \langle x, y \rangle \in E,$ $\mathbf{d} \prec \mathbf{e}$ or $\mathbf{d} \succ \mathbf{e}, R_f \in \mathcal{L}_d(\langle x, y \rangle),$ $R_f \notin \mathcal{L}_e(\langle x, y \rangle)$ then $\mathcal{L}_e(\langle x, y \rangle) := \mathcal{L}_e(\langle x, y \rangle) \cup \{R_f\}$
\forall -rule: if $x \in V_d, \forall R.C \in \mathcal{L}_d(x),$ and there exists R-successor $y \in V_d$ of x s.t. $C \notin \mathcal{L}_d(y)$ then $\mathcal{L}_d(y) := \mathcal{L}_d(y) \cup \{C\}$	\top_A -rule: if $x \in V_e, A_d \in \mathcal{L}_e(x), \top_d \notin \mathcal{L}_e(x)$ then $\mathcal{L}_e(x) := \mathcal{L}_e(x) \cup \{\top_d\}$
\top -rule: if $x \in V_d, C \sqsubseteq D \in \mathbf{K}(C_d),$ $\text{nnf}(\neg C \sqcup D) \notin \mathcal{L}_d(x)$ then $\mathcal{L}_d(x) := \mathcal{L}_d(x) \cup \{\text{nnf}(\neg C \sqcup D)\}$	\top_R -rule: if $x, y \in V_e, \langle x, y \rangle \in E,$ $R_d \in \mathcal{L}_e(\langle x, y \rangle),$ $\top_d \notin \mathcal{L}_e(x) \cap \mathcal{L}_e(y)$ then $\mathcal{L}_e(x) := \mathcal{L}_e(x) \cup \{\top_d\},$ $\mathcal{L}_e(y) := \mathcal{L}_e(y) \cup \{\top_d\}$
$\Delta\uparrow$ -rule: if $x \in V_d, \mathbf{d} \prec \mathbf{e}, x \notin V_e$ then $V_e := V_e \cup \{x\}$	\top_{\sqsubseteq} -rule: if $x \in V_d, \mathbf{e} \prec \mathbf{f}, \neg \top_e \sqcup \top_f \notin \mathcal{L}_d(x)$ then $\mathcal{L}_d(x) := \mathcal{L}_d(x) \cup \{\neg \top_e \sqcup \top_f\}$
	M-rule: if $a^g \in V_d, a^h \in V_e,$ and $\mathbf{d} \preceq \mathbf{e},$ then $\text{merge}(a^g, a^h)$

Definition 5 (Blocking). Given a CKR \mathfrak{K} and a completion tree $T = \langle V, E, \mathcal{L} \rangle$, we say that a node $w \in V$ is the witness for $x \in V$, if $\mathcal{L}_d(x) = \mathcal{L}_d(w)$ for all $\mathbf{d} \in \mathfrak{D}_\Gamma$, $w <_V x$ and there is no $y \in V$ such that $y <_V w$ and $\mathcal{L}_d(x) = \mathcal{L}_d(y)$ for all $\mathbf{d} \in \mathfrak{D}_\Gamma$. We say that $x \in V$ is blocked by $w \in V$ if w is the witness for x .

We say that a tableaux rule is *applicable* if all of its preconditions (the if-part of the rule) are satisfied for some node $x \in V$ or a pair of nodes $x, y \in V$ and the nodes are not blocked. A completion tree T is *complete*, if none of the tableaux rules is applicable. A completion tree $T = \langle V, E, \mathcal{L} \rangle$ contains a *clash* in a node $x \in V$, if for some $\mathbf{d} \in \mathfrak{D}_\Gamma$ and some concept C both $C \in \mathcal{L}_d(x)$ and $\neg C \in \mathcal{L}_d(x)$, or if $\perp \in \mathcal{L}_d(x)$. We say that T is *clash-free* if no clash occurs in any of its nodes.

In initialization, ABox axioms are encoded in the initial completion tree. This technique is well known for logics like \mathcal{ALC} [1]. However, we must consider that in CKR same individuals appearing in different contexts may possibly have different meanings. In the completion tree, individuals will be represented by elements of the form a^g where $a \in N_I$ and $g \in \mathfrak{D}_\Gamma$ identifies the context in which the individual was first introduced. To implement condition 4 of CKR-models we will merge nodes when needed.

Definition 6 (Merging). Executing $\text{merge}(x, y)$ on a completion tree $T = \langle V, E, \mathcal{L} \rangle$, with $x, y \in V$, transforms T as follows: a) node x is added into V_e for all $\mathbf{e} \in \mathfrak{D}_\Gamma$ s.t. $y \in V_e$; b) all concepts from $\mathcal{L}_e(y)$ are added into $\mathcal{L}_e(x)$, for all $\mathbf{e} \in \mathfrak{D}_\Gamma$; c) all edges directed into/from y are redirected into/from x ; d) node y is removed from V .

Finally, the algorithm is formally defined as follows:

Definition 7 (Algorithm C_T). Given as input a CKR \mathfrak{K} , $\mathbf{d} \in \mathfrak{D}_\Gamma$, and a concept C in NNF, the algorithm C_T verifies the \mathbf{d} -satisfiability of C w.r.t. \mathfrak{K} in the following steps:

1. for all $e \in \mathcal{D}_T$, initialize V_e , E , and \mathcal{L}_e as follows:
 - (a) $V_e := \{a^e \mid C(a) \in \mathcal{K}(\mathcal{C}_e)\} \cup \{a^e, b^e \mid R(a, b) \in \mathcal{K}(\mathcal{C}_e)\}$;
 $E := \{\langle a^e, b^e \rangle \mid R(a, b) \in \mathcal{K}(\mathcal{C}_e), e \in \mathcal{D}_T\}$;
 $\mathcal{L}_e(a^e) := \{C \mid C(a) \in \mathcal{K}(\mathcal{C}_e)\}$; $\mathcal{L}_e(\langle a^e, b^e \rangle) := \{R \mid R(a, b) \in \mathcal{K}(\mathcal{C}_e)\}$;
 - (b) $V_d := V_d \cup \{s_0\}$, where s_0 is a new constant in V_d ; $\mathcal{L}_d(s_0) := \{C\}$;
2. exhaustively apply completion rules of Table 1 on T ;
3. once T is complete, answer “ C is \mathbf{d} -satisfiable w.r.t. \mathfrak{R} ” if T is clash-free; answer “ C is not \mathbf{d} -satisfiable w.r.t. \mathfrak{R} ” otherwise.

The first five rules used by the algorithm (from \sqcap - to \mathcal{T} -rule) are the usual \mathcal{ALC} tableaux rules [1] responsible for local reasoning inside each context. The additional rules are new and they handle propagation of information between contexts.

The $\Delta\uparrow$ - and $\Delta\downarrow$ -rules are responsible for propagation of nodes: if $\mathbf{d} \prec \mathbf{e}$, all nodes from V_d are propagated to V_e ($\Delta\uparrow$ -rule), but only the nodes belonging to \top_d are propagated from V_e to V_d ($\Delta\downarrow$ -rule).

Given contexts \mathcal{C}_d and \mathcal{C}_e , with $\mathbf{d} \prec \mathbf{e}$, the conditions 6 and 7 of CKR-models require that the interpretations of any symbol X_f in the contexts agree on all elements shared by their domains. Hence, if a node (or a pair of nodes) belongs to both V_d and V_e (i.e. it belongs to both local tableaux), then its labels are propagated by A -rule and R -rule from one local tableaux to another, in both directions.

The following rules maintain the first three semantic conditions of CKR-models. The \top_{A^-} - and \top_R -rules take care that any qualified symbol X_d is always roofed under \top_d in any context \mathcal{C}_e . If a qualified concept A_d (role R_d) is found in the \mathcal{L}_e -label of some node (edge) in V_e , then \top_d is added to the \mathcal{L}_e -label of this node (or both nodes connected by this edge). Also, if $\mathbf{e} \prec \mathbf{f}$, then the \top_{\sqsubseteq} -rule assures that the subsumption $\top_e \sqsubseteq \top_f$ must hold in any context. Finally, the M -rule takes care of cases when it is inferred that the same individual a appears in two different contexts.

It is however not the case that there is one-to-one correspondence between the semantic conditions of CKR (Definition 3) and the tableaux rules. Consider condition 6 and the case when $X_d = A_d$ and $\mathbf{d} \prec \mathbf{e}$. If for instance due to a firing of the \exists -rule a new node x was added into V_e with $\mathcal{L}_e(x)$ initiated to $\{A_d\}$, to maintain condition 6 the same node with the same label must also be added to V_d and \mathcal{L}_d respectively. This is achieved by consecutive firing of \top_{A^-} , $\Delta\downarrow$ -, and A -rules. A more complex example of reasoning with CKR tableaux rules follows.

Example 1 (Tableaux algorithm). Using the algorithm and our example CKR \mathfrak{R}_{fb} , let us show the proof for the following subsumption:

$$\mathfrak{R}_{fb} \models \mathbf{nfl10} : \text{WorldChampionPlayer}_{fb} \sqsubseteq \forall \text{playsFor}_{wc10}. \text{WinnerTeam}_{wc10}$$

Initialization yields $V_{\mathbf{nfl10}} := \{s_0\}$ and $\mathcal{L}_{\mathbf{nfl10}}(s_0) := \{\text{WorldChampionPlayer}_{fb} \sqcap \exists \text{playsFor}_{wc10}. \neg \text{WinnerTeam}_{wc10}\}$. Then tableaux rules are applied as follows:

- (1) $\mathcal{L}_{\mathbf{nfl10}}(s_0) := \mathcal{L}_{\mathbf{nfl10}}(s_0) \cup \{\text{WorldChampionPlayer}_{fb}, \exists \text{playsFor}_{wc10}. \neg \text{WinnerTeam}_{wc10}\}$ by \sqcap -rule;
- (2) $V_{\mathbf{nfl10}} := V_{\mathbf{nfl10}} \cup \{s_1\}$, $E_{\mathbf{nfl10}} := \{\langle s_0, s_1 \rangle\}$,
 $\mathcal{L}_{\mathbf{nfl10}}(\langle s_0, s_1 \rangle) := \{\text{playsFor}_{wc10}\}$, $\mathcal{L}_{\mathbf{nfl10}}(s_1) := \{\neg \text{WinnerTeam}_{wc10}\}$ by \exists -rule;
- (3) $V_{fb} := \{s_0, s_1\}$, $\mathcal{L}_{fb}(s_0) := \{\text{WorldChampionPlayer}\}$,
 $\mathcal{L}_{fb}(\langle s_0, s_1 \rangle) := \{\text{playsFor}_{wc10}\}$ by $\Delta\uparrow$ -, A - and R -rules;

- (4) $\mathcal{L}_{fb}(s_0) \cup \{\text{ChampionPlayer}_{wc10}\}$ by \mathcal{T} - and \sqcup -rules;
- (5) $\mathcal{L}_{fb}(s_0) := \mathcal{L}_{fb}(s_0) \cup \{\top_{wc10}\}$, $\mathcal{L}_{fb}(s_1) := \mathcal{L}_{fb}(s_1) \cup \{\top_{wc10}\}$ by \top_R -rule;
- (6) $V_{wc10} := \{s_0, s_1\}$, $\mathcal{L}_{wc10}(s_0) := \{\text{ChampionPlayer}\}$,
 $\mathcal{L}_{wc10}(\langle s_0, s_1 \rangle) := \{\text{playsFor}\}$ by $\Delta\downarrow$ -, A - and R -rules;
- (7) $\mathcal{L}_{wc10}(s_0) := \mathcal{L}_{wc10}(s_0) \cup \{\forall \text{playsFor.WinnerTeam}\}$ by \mathcal{T} - and \sqcup -rules;
- (8) $\mathcal{L}_{wc10}(s_1) := \mathcal{L}_{wc10}(s_1) \cup \{\text{WinnerTeam}\}$ by \forall -rule;
- (9) $\mathcal{L}_{fb}(s_1) := \mathcal{L}_{fb}(s_1) \cup \{\text{WinnerTeam}_{wc10}\}$,
 $\mathcal{L}_{nfl10}(s_1) := \mathcal{L}_{nfl10}(s_1) \cup \{\text{WinnerTeam}_{wc10}\}$ by A -rule;

The application of last rule creates a clash, since $\mathcal{L}_{nfl10}(s_1) = \{\neg \text{WinnerTeam}_{wc10}, \text{WinnerTeam}_{wc10}\}$. Note that in the non-deterministic choices asked in steps 4 and 7 (due to \sqcup -rule), all other choices immediately lead to a clash. Hence no clash-free completion tree can be constructed and the algorithm answers that the input concept is **nfl10**-unsatisfiable w.r.t. \mathfrak{K}_{fb} . This implies that the subsumption in question is entailed.

Note the required inter-contextual knowledge propagation: we first had to propagate nodes and their labels from V_{nfl10} to V_{fb} and finally to V_{wc10} by tracking the context coverage structure (steps 3–6). Then with the last rule application (step 9), we propagate back the derived concepts to the label \mathcal{L}_{nfl10} and detect the clash. \diamond

The algorithm $C_{\mathcal{T}}$ is correct: it terminates on any input and it is sound and complete.

Theorem 1 (Correctness). *Given a CKR \mathfrak{K} , $\mathbf{d} \in \mathfrak{D}_{\Gamma}$, and a concept C in NNF on the input, the tableaux algorithm $C_{\mathcal{T}}$ always terminates and C is \mathbf{d} -satisfiable w.r.t. \mathfrak{K} iff $C_{\mathcal{T}}$ generates a complete and clash free completion tree.*

The \mathcal{ALC} tableaux algorithm which we extended in this paper is in NEXPTIME [6], and this is the case also for the resulting tableaux algorithm for CKR.

Theorem 2 (Complexity). *The complexity of the $C_{\mathcal{T}}$ algorithm is NEXPTIME with respect to the combined size of the input.*

In general, the problem of deciding \mathbf{d} -satisfiability (and thus \mathbf{d} -subsumption) in \mathcal{ALC} -based CKR is EXPTIME-complete [4]. That is, the complexity is the same as for \mathcal{ALC} with general TBoxes [1]. To obtain an optimal algorithm for CKR on top of \mathcal{ALC} we would have to extend one of the EXPTIME algorithms for \mathcal{ALC} (like, e.g., [7]). On the other hand, the algorithm presented in this paper is an important first step towards the algorithmic support for CKR based on more expressive DL (like \mathcal{SHIQ} or \mathcal{SROIQ}), since the tableaux algorithms for these logics can be seen as extensions of the basic \mathcal{ALC} algorithm on top of which we have built.

4 Algorithm Optimization

In this section we share our initial ideas about optimization of the algorithm. CKR maintains a certain level of separation between meta and object knowledge (the former influences the latter but not vice versa). Object reasoning queries the meta knowledge only to verify the coverage between dimensional vectors. The number of contexts m is typically much smaller than the size of whole KB n and the number of dimensions k

is assumed to be a constant, it hence makes sense to precompute³ the context coverage beforehand. This can be done within $k \times m^2 = O(m^2)$ queries of the form $\mathfrak{M} \models d \prec_A d'$. Consequently, meta reasoning does not slow down object reasoning more than in other approaches with simpler meta knowledge representations [14].

One of the advantages of contextual reasoning is that the KB is split into smaller units and reasoning can be parallelized. Let us briefly sketch how this can be done with CKR. Reasoning in each context will be handled by a separate processor, which will exchange messages to deal with knowledge propagation. The computation time will be bounded by the context which requires the longest execution time together with the number of required messages. In this sense, the \sqcup -, \sqcap -, \exists -, \forall -, \top_A -, \top_R -, and \top_{\square} -rules are locally executed. The remaining rules will be implemented as follows:

$\Delta\uparrow$ -, $\Delta\downarrow$ -rules: the fact that a node has to be added into the target context is detected locally in the source context. A message is sent into the target context and this fact is also cached in the source context, which will be used by the other rules.

A -, R -rules: thanks to caching of the information to which contexts nodes have been added, it can be locally detected that the concept and role labels of some node have to be propagated to the target context which is then done by a message.

M -rule: note that if $a^g \in V_d$, $a^h \in V_e$, and $d \prec e$, then eventually a^g is added into V_e by the $\Delta\uparrow$ -rule. Therefore also the precondition the M -rule can be locally verified and once detected, a respective message is sent to all other contexts.

Propagation of knowledge increases the number of messages and can trigger additional computation in the target context. It is hence desired to limit it to the necessary cases only. Using a technique similar to *lazy unfolding* [1], we were able to optimize the three tableaux rules \top_A -, \top_R -, and \top_{\square} for propagation of the \top_e symbols as follows:

$$\begin{aligned} \top_A^* \text{-rule:} \quad & \text{if } x \in V_f, \mathbf{d} \preceq \mathbf{e}, A_d \in \mathcal{L}_f(x), \top_e \notin \mathcal{L}_f(x) \\ & \text{then } \mathcal{L}_f(x) := \mathcal{L}_f(x) \cup \{\top_e\} \\ \top_R^* \text{-rule:} \quad & \text{if } x, y \in V_f, \mathbf{d} \preceq \mathbf{e}, R_d \in \mathcal{L}_f(\langle x, y \rangle), \top_e \notin \mathcal{L}_f(x) \cap \mathcal{L}_f(y) \\ & \text{then } \mathcal{L}_f(x) := \mathcal{L}_f(x) \cup \{\top_e\}, \mathcal{L}_f(y) := \mathcal{L}_f(y) \cup \{\top_e\} \\ \top_{\square}^* \text{-rule:} \quad & \text{if } x \in V_f, \mathbf{d} \prec \mathbf{e}, \neg \top_e \in \mathcal{L}_f(x), \neg \top_d \notin \mathcal{L}_f(x) \\ & \text{then } \mathcal{L}_f(x) := \mathcal{L}_f(x) \cup \{\neg \top_d\} \end{aligned}$$

The main idea of these optimized rules is to avoid the introduction of a number of disjunctive concept expressions of the form $\neg \top_d \sqcup \top_e$ caused by the \top_{\square} -rule which could possibly cause unnecessary non-deterministic branching. Instead, we apply each disjunction only after one of the disjuncts is proven untrue.

Normally the \top_A -rule would add \top_d into the label of any node x in which A_d was found. Consequently, the \top_{\square} -rule would be fired once for each $e \succ d$ and add $\neg \top_d \sqcup \top_e$ every time. This eventually results into adding \top_e into the same label for each such e over the run of the algorithm. The optimized \top_A^* -rule skips the introduction of these disjunctions and directly adds the \top_e symbol for all such e . The \top_R -rule is also optimized in the very same fashion. Hence the two optimized rules \top_A^* - and \top_R^* -rule

³ This does not imply that a DL KB at meta level is useless. In meta knowledge modeling, DL axioms on dimensional values can constrain the coverage structure, e.g., given a location dimension, we can require cities to be located within some country: $\text{City} \sqsubseteq \exists \prec_{\text{location}} \text{Country}$.

do the work previously done by the \top_{A^-} - and \top_{R^-} -rules but in addition they take care of the first part of the disjunction $\neg\top_{\mathbf{d}} \sqcup \top_{\mathbf{e}}$ (i.e., the one which adds $\top_{\mathbf{e}}$ if $\top_{\mathbf{d}}$ was found). We still have to take care of the second part, and this is done by the $\top_{\underline{\mathbf{e}}}^*$ -rule which adds $\neg\top_{\mathbf{d}}$ to any label in which $\neg\top_{\mathbf{e}}$ was found for $\mathbf{d} \prec \mathbf{e}$.

The version of the algorithm $C_{\mathcal{T}}$ that uses the \top_{A^-} -, \top_{R^-} -, and $\top_{\underline{\mathbf{e}}}^*$ -rules instead of the \top_{A^-} -, \top_{R^-} -, and $\top_{\underline{\mathbf{e}}}$ -rules respectively, will be denoted by $C_{\mathcal{T}}^*$.

Theorem 3 (Correctness of optimized rules). *Given a CKR \mathfrak{K} , $\mathbf{d} \in \mathfrak{D}_{\Gamma}$, and a concept C in NNF on the input, the algorithm $C_{\mathcal{T}}^*$ always terminates and C is \mathbf{d} -satisfiable w.r.t. \mathfrak{K} iff $C_{\mathcal{T}}^*$ generates a complete and clash free completion tree.*

Example 2 (Optimized tableaux rules). Let us now compare the original tableaux rules with the optimized rules by the following deduction:

$$\mathfrak{K}_{fb} \models \text{sp} : \text{TopSportsman} \sqsubseteq \forall \text{playsFor}_{\text{wc10}}. \text{WinnerTeam}_{\text{wc10}}$$

The algorithm is initialized with $V_{\text{sp}} = \{s_0\}$ and the label $\mathcal{L}_{\text{sp}} = \{\text{TopSportsman} \sqcap \exists \text{playsFor}_{\text{wc10}}. \neg \text{WinnerTeam}_{\text{wc10}}\}$. The original algorithm $C_{\mathcal{T}}$ proceeds as follows:

- (1) $\mathcal{L}_{\text{sp}}(s_0) := \mathcal{L}_{\text{sp}}(s_0) \cup \{\text{TopSportsman}, \exists \text{playsFor}_{\text{wc10}}. \neg \text{WinnerTeam}_{\text{wc10}}\}$ by \sqcap -rule;
- (2) $V_{\text{sp}} := V_{\text{sp}} \cup \{s_1\}$, $E_{\text{sp}} = \{\langle s_0, s_1 \rangle\}$, $\mathcal{L}_{\text{sp}}(\langle s_0, s_1 \rangle) = \{\text{playsFor}_{\text{wc10}}\}$,
 $\mathcal{L}_{\text{sp}}(s_1) = \{\neg \text{WinnerTeam}_{\text{wc10}}\}$ by \exists -rule;
- (3) $\mathcal{L}_{\text{sp}}(s_0) := \mathcal{L}_{\text{sp}}(s_0) \cup \{\top_{\text{wc10}}\}$, $\mathcal{L}_{\text{sp}}(s_1) := \mathcal{L}_{\text{sp}}(s_1) \cup \{\top_{\text{wc10}}\}$ by \top_{R^-} -rule;
- (4) $\mathcal{L}_{\text{sp}}(s_0) := \mathcal{L}_{\text{sp}}(s_0) \cup \{\neg\top_{\text{wc10}} \sqcup \top_{\text{fb}}, \neg\top_{\text{nfl10}} \sqcup \top_{\text{fb}}, \neg\top_{\text{wc10}} \sqcup \top_{\text{sp}}, \neg\top_{\text{nfl10}} \sqcup \top_{\text{sp}}, \neg\top_{\text{fb}} \sqcup \top_{\text{sp}}\}$,
 $\mathcal{L}_{\text{sp}}(s_1) := \mathcal{L}_{\text{sp}}(s_1) \cup \{\neg\top_{\text{wc10}} \sqcup \top_{\text{fb}}, \neg\top_{\text{nfl10}} \sqcup \top_{\text{fb}}, \neg\top_{\text{wc10}} \sqcup \top_{\text{sp}}, \neg\top_{\text{nfl10}} \sqcup \top_{\text{sp}}, \neg\top_{\text{fb}} \sqcup \top_{\text{sp}}\}$ by multiple applications of the $\top_{\underline{\mathbf{e}}}$ -rule;
- (5) $\mathcal{L}_{\text{sp}}(s_0) := \mathcal{L}_{\text{sp}}(s_0) \cup \{\top_{\text{fb}}, \top_{\text{sp}}\}$, $\mathcal{L}_{\text{sp}}(s_1) := \mathcal{L}_{\text{sp}}(s_1) \cup \{\neg\top_{\text{nfl10}}, \top_{\text{fb}}, \top_{\text{sp}}\}$ by \sqcup -rule;
- (6) $V_{\text{fb}} = \{s_0, s_1\}$, $\mathcal{L}_{\text{fb}}(s_0) = \{\text{TopSportsman}_{\text{sp}}\}$ by $\Delta\downarrow$ - and A -rules;
- (7) $\mathcal{L}_{\text{fb}}(s_0) := \mathcal{L}_{\text{fb}}(s_0) \cup \{\text{WorldChampionPlayer}\}$ by \mathcal{T} -rule⁴ and \sqcup -rule;
- (8) $\mathcal{L}_{\text{fb}}(s_0) := \mathcal{L}_{\text{fb}}(s_0) \cup \{\text{ChampionPlayer}_{\text{wc10}}\}$ by \mathcal{T} - and \sqcup -rules;
- (9) $V_{\text{wc10}} = \{s_0, s_1\}$, $\mathcal{L}_{\text{wc10}}(s_0) = \{\text{ChampionPlayer}\}$, $\mathcal{L}_{\text{wc10}}(\langle s_0, s_1 \rangle) = \{\text{playsFor}\}$ by $\Delta\downarrow$ - A and R -rules;
- (10) $\mathcal{L}_{\text{wc10}}(s_0) := \mathcal{L}_{\text{wc10}}(s_0) \cup \{\forall \text{playsFor}. \text{WinnerTeam}\}$ by \mathcal{T} - and \sqcup -rules;
- (11) $\mathcal{L}_{\text{wc10}}(s_1) := \mathcal{L}_{\text{wc10}}(s_1) \cup \{\text{WinnerTeam}\}$ by \forall -rule;
- (12) $\mathcal{L}_{\text{sp}}(s_1) := \mathcal{L}_{\text{sp}}(s_1) \cup \{\text{WinnerTeam}_{\text{wc10}}\}$ by A -rule;

The last rule application yields a clash since we obtain $\mathcal{L}_{\text{sp}}(s_1) = \{\neg \text{WinnerTeam}_{\text{wc10}}, \text{WinnerTeam}_{\text{wc10}}\}$. Notice that out of the ten applications of the $\top_{\underline{\mathbf{e}}}$ -rule in step (4), only the one resulting into adding $\neg\top_{\text{wc10}} \sqcup \top_{\text{fb}}$ into $\mathcal{L}_{\text{fb}}(s_0)$ is actually needed for propagation of the concept TopSportsman into $\mathcal{L}_{\text{fb}}(s_0)$. On the other hand, the addition of $\neg\top_{\text{nfl10}} \sqcup \top_{\text{fb}}, \neg\top_{\text{nfl10}} \sqcup \top_{\text{sp}}$ into the labels of both nodes (carrying irrelevant information about the context for nfl10) is preliminary at this point and it may lead to unnecessary choices by the \sqcup -rule which may need to be backtracked later on – for instance the choice to add $\neg\top_{\text{nfl10}}$ to s_1 in step (5). If instead the optimized algorithm $C_{\mathcal{T}}^*$ is used, a similar derivation is obtained in which steps (3)–(5) are replaced with:

⁴ The two disjunctive concepts $\neg\text{TopSportsman}_{\text{sp}} \sqcup \text{WorldChampionPlayer}$ and $\neg\text{WorldChampionPlayer} \sqcup \text{ChampionPlayer}_{\text{wc10}}$ which are added to \mathcal{L}_{fb} in steps (7) and (8) respectively by the \mathcal{T} -rule are not listed here to improve readability.

$$(3^*) \begin{aligned} \mathcal{L}_{\text{sp}}(s_0) &:= \mathcal{L}_{\text{sp}}(s_0) \cup \{\top_{\text{wc10}}, \top_{\text{fb}}, \top_{\text{sp}}\}, \\ \mathcal{L}_{\text{sp}}(s_1) &:= \mathcal{L}_{\text{sp}}(s_1) \cup \{\top_{\text{wc10}}, \top_{\text{fb}}, \top_{\text{sp}}\} \text{ by } \top_R^* \text{-rule;} \end{aligned}$$

The remainder of derivation is the same: the unnecessary choice is thus avoided. \diamond

The optimized rules constrain the propagation of \top_e concepts, which are needed to reflect the context hierarchy in reasoning, to necessary propagations only and avoid the introduction of unnecessary disjunctive concepts which may cause branching. Observe that in Examples 1 and 2 we have shown the application of rules in the right order. However, additional non relevant rules may be applied by the algorithm before a clash is reached. For instance, due to the axiom $\text{WinnerTeam} \equiv \text{WinnerFinal}$ in $\mathcal{K}(\mathcal{C}_{\text{wc10}})$ the algorithm may add WinnerFinal into $\mathcal{L}_{\text{wc10}}(s_1)$ after step (11) in Example 2 (by \mathcal{T} - and \sqcup -rules) and consequently propagate $\text{WinnerFinal}_{\text{wc10}}$ into $\mathcal{L}_{\text{fb}}(s_1)$ and $\mathcal{L}_{\text{sp}}(s_1)$ by A -rule. Such a propagation is unnecessary as there are no axioms in \mathcal{C}_{fb} nor \mathcal{C}_{sp} which could derive new knowledge from $\text{WinnerFinal}_{\text{wc10}}$. Therefore in the future we would also like to investigate when it is necessary to propagate qualified symbols.

5 Related Works

The only other approach for reasoning with DL-based CKR is a translation from CKR into a single DL KB [13]. Unfortunately, this solution is not practically efficient, as the translation adds a large number of axioms in order to track complex relations between qualified symbols in a single KB. This is reflected also by a significant (cubic) blow up in the size of knowledge base after the translation. In contrast, a direct tableaux algorithm allows for more effective reasoning: local reasoning is executed in the respective part of the completion tree and only relevant consequences posed on other contexts are propagated into their respective tableaux labels, thus opening the possibility of parallelization. Our tableaux procedure is also related to the distributed tableaux algorithms for DDL [8] and P-DL [2], especially in the way how symbols are propagated between local tableaux. Apart from the fact that each of these algorithms implements a different semantics, our algorithm is also able to handle semantic dependencies between roles which is an open problem for DDL and P-DL so far.

Our newly introduced optimizations bring us near to approaches interested in parallelization of DL reasoning. One relevant approach in this area is presented in [10]. This work proposes a saturation procedure for the classification of the polynomial fragment $\mathcal{ELH}_{\mathcal{R}^+}$ of OWL 2 EL, distributable among multiple processors as a concurrent algorithm. The paper also presents an implementation in the reasoner ELK together with a promising evaluation over known \mathcal{EL} ontologies. Even if the scope of [10] is different from our work, it highlights some aspects that support our approach. In particular, it shows that there is interest in a parallelized vision of DL reasoning algorithms. Moreover, it suggests that the sort of knowledge distribution and independence between contexts which we point to can effectively result in promising performance improvements.

6 Conclusions

Contextualized Knowledge Repository (CKR) is a knowledge representation framework that provides a contextual layer for DL knowledge bases. The recently introduced

reasoning algorithm [5, 9] for \mathcal{ALC} -based CKR provides the first direct tableaux decision procedure for contextualized knowledge. This solution is more effective than the previously known approaches based on reduction that lead to KB blow ups and loss of the divide-and-conquer advantage of contextual representation.

In this paper, we reviewed the algorithm and discussed on its possible optimization including dimensional structure caching, parallelization and a set of new rules that optimize the propagation of symbols among local tableaux. In the future we want to extend the algorithm towards more expressive DL such as \mathcal{SHIQ} and \mathcal{SROIQ} and formulate an EXPTIME algorithm based on the existing approaches [7]: we note that some of the optimizations (e.g. the lazy unfolding for \top_{\square} or the precomputation of context coverage) can be easily adapted to different formulations of the algorithm. We will also study further optimizations for the propagation of qualified symbols.

Acknowledgements: This research was supported from the LiveMemories project. Martin Homola is also supported from the Slovak national project VEGA no. 1/1333/12.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The description logic handbook. Cambridge University Press (2003)
2. Bao, J., Caragea, D., Honavar, V.: A distributed tableau algorithm for package-based description logics. In: CRR 2006 (2006)
3. Bao, J., Tao, J., McGuinness, D.L.: Context representation for the semantic web. In: WebSci10 (2010)
4. Bozzato, L., Homola, M., Serafini, L.: ExpTime reasoning for contextualized \mathcal{ALC} . Tech. Rep. TR-FBK-DKM-2012-1, Fondazione Bruno Kessler, Trento, Italy (2012), <http://dkm.fbk.eu/index.php/Resources>
5. Bozzato, L., Homola, M., Serafini, L.: Tableaux for contextualized description logics. Submitted (2012)
6. Buchheit, M., Donini, F.M., Schaerf, A.: Decidable reasoning in terminological knowledge representation systems. J. Artif. Intell. Res. (JAIR) 1, 109–138 (1993)
7. Goré, R., Nguyen, L.: EXPTIME tableaux for \mathcal{ALC} using sound global caching. In: DL2007. CEUR-WP, vol. 250, pp. 299–306. CEUR-WS.org (2007)
8. Homola, M., Serafini, L.: Augmenting subsumption propagation in distributed description logics. App. Artif. Intell. 24(1-2), 137–174 (2010)
9. Homola, M., Bozzato, L., Serafini, L.: Tableaux algorithm for reasoning with contextualized knowledge. Tech. Rep. TR-FBK-DKM-2011-1, Fondazione Bruno Kessler, Trento, Italy (2011), <http://dkm.fbk.eu/index.php/Resources>
10. Kazakov, Y., Krötzsch, M., Simancik, F.: Concurrent classification of \mathcal{EL} ontologies. In: ISWC 2011. LNCS, vol. 7031, pp. 305–320. Springer (2011)
11. Klarman, S., Gutiérrez-Basulto, V.: Two-dimensional description logics for context-based semantic interoperability. In: AAAI-11. AAAI Press (2011)
12. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Art. Int. 48(1), 1–26 (1991)
13. Serafini, L., Homola, M.: Contextualized knowledge repositories for the semantic web. J. of Web Sem., Special Issue: Reasoning with context in the Semantic Web 12 (2012)
14. Straccia, U., Lopes, N., Lukacsy, G., Polleres, A.: A general framework for representing and reasoning with annotated Semantic Web data. In: AAAI-10. AAAI Press (2010)

Inverting Subsumption for Constructive Reasoning

Simona Colucci, Francesco M. Donini

DISUCOM, Università della Tuscia, Viterbo, Italy

Abstract. We present a Logic Programming prototype implementation, working as proof-of-concept for a unified strategy proposed in our past research to solve several non-standard reasoning problems in Description Logics (DLs), denoted by *Constructive Reasoning*. In order to prove both the problem-independence and the logic-independence of the adopted approach, the prototype is focused on the solution of three different problems — namely Least Common Subsumer, Concept Abduction and Concept Difference — and two different, though simple and endowed with structural subsumption, DLs, *i.e.*, \mathcal{EL} and \mathcal{ALN} . Accordingly to the implemented strategy, problems are formalized as conjunction of both subsumption and non-subsumption statements, causing the whole prototype to rely on a Prolog program solving subsumption. The program is built around a predicate, which on the one hand checks for the existence of subsumption relations between ground elements, providing boolean answers, and on the other hand, if inverted, exploits Prolog built-in unification to enumerate variable values making subsumption true between concept terms containing concept variables.

1 Introduction

The power of knowledge lays in its ability to enhance the production of unknown information, through management strategies whose significance increases with the level of novelty introduced by provided results.

In past knowledge management literature, in fact, interest has been given to the proposal of special purpose inferences allowing for exploiting as much as possible the informative content achieved through knowledge representation effort. To this aim, several non-standard reasoning services have been proposed and continue to be investigated to cope with different representation or inference needs. The most relevant services we may cite are explanation [16], interpolation [18], concept abduction [12], concept contraction [11], concept unification [5], concept difference [19], concept similarity [8], concept rewriting [3], least common subsumer [7], most specific concept [1], knowledge base completion [6], forgetting or uniform interpolation [14].

We notice that the crucial role of non-standard reasoning in the process of capturing unexpected sources of information has been stressed also in research fields apparently far from knowledge representation [15].

Moreover, recent Description Logics (DLs) literature has shown interest for easily tractable, even though not very expressive, sub-languages, like \mathcal{EL} [17, 4, 13].

In our past research [10] we proposed an integrated approach and solving strategy for dealing with several different non-standard inferences. The framework, presented as independent of the DL adopted for knowledge representation, takes a *constructive*

reasoning perspective on problem solving: most inferences are in the form “Find one or more concept(s) C such that {sentence involving C }“ and the proposed framework aims at building such C .

In order to show the feasibility of such an integrated constructive reasoning approach, we here present a prototype implementation in Logic Programming solving Least Common Subsumer, Concept Difference and Concept Abduction in the simple DLs \mathcal{EL} , \mathcal{ALN} , both endowed with structural subsumption algorithms.

Though still inefficient at this stage, the prototype works as proof-of-concept for the integrated solution framework. It exploits the property of our approach according to which most non-standard reasoning problems may be formalized as conjunction of both subsumption and non-subsumption statements and therefore relies on a Prolog program solving subsumption, built around a main predicate called either *subs_el* or *subs_aln*, depending on the adopted \mathcal{DL} . In particular, we show how to invert the *subs* predicate (either *subs_el* or *subs_aln*), so that not only it can check subsumption between ground elements (providing boolean answers), but it can also exploit Prolog built-in unification to enumerate variable values making subsumption true between concept terms containing concept variables. The approach takes a generate-and-test strategy.

In the next section, we shortly recall how to formalize the three problems in the integrated framework. Then, we describe the architecture of the prototype implementing the solving strategy in Section 3, before delving into details of subsumption program, on which the whole prototype relies, in Section 4. We show how to query the presented prototype in Section 5, and, finally, close the paper with discussions and future work.

2 Background Framework

The approach presented in our past research [10] models each of the problems at hand as Optimal Solution Problem, whose definition exploits specific second order formulas, written as conjunction of concept subsumptions and non-subsumptions, in the following form:

$$\Gamma = (C_1 \sqsubseteq D_1) \wedge \cdots \wedge (C_\ell \sqsubseteq D_\ell) \wedge (C_{\ell+1} \not\sqsubseteq D_{\ell+1}) \wedge \cdots \wedge (C_m \not\sqsubseteq D_m) \quad (1)$$

In Formula (1), $C_1, \dots, C_m, D_1, \dots, D_m \in \mathcal{DL}$ denote concept terms containing concept variables X_0, X_1, \dots, X_n . We say that Γ is satisfiable in DL iff there exists a substitution $\sigma = X_0 \mapsto E_0, \dots, X_n \mapsto E_n$ such that $\sigma(\Gamma)$ is true (i.e., each subsumption and non-subsumption statement in (1) is true). If Γ is satisfiable in \mathcal{DL} then \mathcal{E} is called a **solution** for Γ and the set of solutions for Γ is defined as:

$$SOL(\Gamma) = \{\mathcal{E} = \langle E_0, \dots, E_n \rangle \mid \mathcal{E} \text{ is a solution for } \Gamma\}$$

Definition 1 (OSP). An Optimal Solution Problem (OSP) \mathbf{P} is a pair $\langle \Gamma, \prec \rangle$, where Γ is a formula of the form (1) and \prec is a preorder over $SOL(\Gamma)$. A solution to \mathbf{P} is a concept tuple \mathcal{E} such that both $\mathcal{E} \in SOL(\Gamma)$ and there is no $\mathcal{E}' \in SOL(\Gamma)$ with $\mathcal{E}' \prec \mathcal{E}$.

2.1 Non-standard Services in DLs as OSPs

In the following, we recall how to model the three investigated problems as OSP. Aiming at a fixpoint computation for solving each of the problems below, a greatest element (*i.e.*, a least preferred one) w.r.t. \prec is provided, which could be used to start the iteration of an inflationary operator.

Least Common Subsumer

Definition 2. [9] *Let C_1 and C_2 be two concepts. The Least Common Subsumer (LCS) of C_1, C_2 is the least element w.r.t. \sqsubseteq of the set of concepts which are Common Subsumers of C_1, C_2 and is unique up to equivalence.*

Common subsumers of C_1, C_2 satisfy the formula of the form (1):

$$\Gamma_{LCS} = (C_1 \sqsubseteq X) \wedge (C_2 \sqsubseteq X)$$

Then, the LCS problem can be expressed by the OSP $LCS = \langle \Gamma_{LCS}, \sqsupseteq \rangle$. We note that \top is always a solution of Γ_{LCS} which is a greatest element w.r.t. \sqsupseteq .

Concept Difference Following the algebraic approaches adopted in classical information retrieval, Concept Difference [19] was introduced as a way to measure concept similarity.

Definition 3. [19] *Let C and D be two concepts such that $C \sqsubseteq D$. The Concept Difference $C - D$ is defined by $\max_{\sqsubseteq} \{B \in \mathcal{DL} \text{ such that } D \sqcap B \equiv C\}$.*

We can define the following formula of the form (1):

$$\Gamma_{DIFF} = (C \sqsubseteq (D \sqcap X)) \wedge ((D \sqcap X) \sqsubseteq C)$$

Such a definition causes Concept Difference to be modeled as the OSP $DIFF = \langle \Gamma_{DIFF}, \sqsupseteq \rangle$. We recall that, in spite of its name, a Concept Difference problem may have several solutions [19]. Note that a greatest solution for Γ_{DIFF} w.r.t. \sqsupseteq is C itself.

Concept Abduction Concept Abduction is a straight adaptation of Propositional Abduction.

Definition 4. [12] *Let C, D , be two concepts in \mathcal{DL} , both C and D satisfiable. A Concept Abduction Problem (CAP) is finding a concept $H \in \mathcal{DL}$ such that $C \sqcap H \not\sqsubseteq \perp$, and $C \sqcap H \sqsubseteq D$.*

Every solution H of a CAP satisfies the formula

$$\Gamma_{ABD} = (C \sqcap X \not\sqsubseteq \perp) \wedge (C \sqcap X \sqsubseteq D)$$

The preference relation for evaluating solutions is subsumption-maximality, since less specific solutions should be preferred because they hypothesize the least. According to the proposed framework, we can model Subsumption-maximal Concept Abduction as $ABD = \langle \Gamma_{ABD}, \sqsupseteq \rangle$. Note that a greatest—*i.e.*, most specific—solution of ABD w.r.t. \sqsupseteq is D , if $C \sqcap D$ is a satisfiable concept (if it is not, then ABD has no solution at all [12, Prop.1]).

2.2 Optimality by Fixpoint

Optimal solutions w.r.t. a preorder might be reached by iterating an inflationary operator. We now specialize the definition of inflationary operators and fixpoints to our setting.

Definition 5 (Inflationary operators and fixpoints). *Given an OSP $\mathbf{P} = \langle \Gamma, \prec \rangle$, we say that the operator $b_{\mathbf{P}} : SOL(\Gamma) \rightarrow SOL(\Gamma)$ (for **better**) is inflationary if for every $\mathcal{E} \in SOL(\Gamma)$, it holds that $b_{\mathbf{P}}(\mathcal{E}) \prec \mathcal{E}$ if \mathcal{E} is not a least element of \prec , $b_{\mathbf{P}}(\mathcal{E}) = \mathcal{E}$ otherwise. In the latter case, we say that \mathcal{E} is a fixpoint of $b_{\mathbf{P}}$.*

Intuitively, $b_{\mathbf{P}}(\mathcal{E})$ is a solution better than \mathcal{E} w.r.t. \prec , if such a solution exists, otherwise a fixpoint has been reached, and such a fixpoint is a solution to \mathbf{P} . Being $b_{\mathbf{P}}$ inflationary, a fixpoint is always reached—possibly in an infinite number of steps—by the following induction: starting from a solution \mathcal{E} , let

$$\begin{aligned} \mathcal{E}_0 &= \mathcal{E} \\ \mathcal{E}_{i+1} &= b_{\mathbf{P}}(\mathcal{E}_i) \text{ for } i = 0, 1, 2, \dots \end{aligned}$$

Then, there exists a limit ordinal λ such that \mathcal{E}_λ is a fixpoint of $b_{\mathbf{P}}$. For each of the previous non-standard reasoning services, we highlighted a greatest solution $\mathcal{E} \in SOL(\Gamma)$ which this iteration can start from. Obviously, when \prec is well-founded (in particular, when $SOL(\Gamma)$ is finite) the fixpoint is reached in a finite number of steps, but the general conditions for well-foundedness of \prec are not known, and out of the scope of this paper. However, also when after n iterations \mathcal{E}_n is not a fixpoint, one can stop and consider \mathcal{E}_n as an approximation of an optimal solution, since $\mathcal{E}_{i+1} \prec \mathcal{E}_i$ for every $i = 0, \dots, n$. In this sense, our method can be used as an anytime approximation.

Note also that the Tarski-Knaster results about uniqueness of the least fixpoint for a monotone operator are not applicable in this setting, first of all, because $b_{\mathbf{P}}$ is not monotone, and secondly because there can be more than one minimal fixpoint: in fact, it is known that for \mathcal{ALN} , both Concept Difference and Concept Abduction admit more than one solution.

We stress the fact that we are *not* proving here that every instance of Formula (1) can be solved by this method. For instance, deciding whether a formula of the form (1) is satisfiable is an open problem for \mathcal{ALN} , to the best of our knowledge. In this paper we address particular cases of (1), corresponding to known non-standard inferences, for which a solution is always known to exist.

It is interesting to observe that such particular cases are similar to *matching* problems [2], in that variables appear only on one side of each subsumption and non-subsumption statement.

3 Prototype Architecture

In the following, we present a prototype Logic Programming system implementing the above mentioned approach to non-standard inference [10]. The system has been developed exploiting the integrated environment provided by SWI-Prolog¹ (Multi-threaded, 32 bits, Version 5.6.64) and follows the modular architecture depicted in Figure 1.

¹ <http://www.swi-prolog.org/>

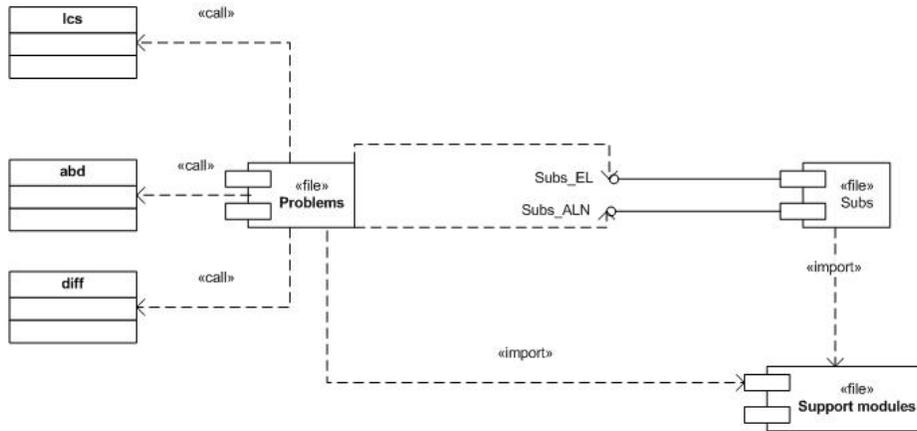


Fig. 1. Prototype Architecture

The system design has been focused on proving main distinguishing features of our approach: the generality and the independence of the adopted DL (within a given subset) of non-standard inferences solving strategy. In particular, the prototype here presented is devoted to the solution of three different reasoning services, namely Least Common Subsumer, Concept Difference and Concept Abduction, in \mathcal{EL} and \mathcal{ALN} .²

Coherently with the strategy introduced so far, the prototype searches for solutions for the system of the OSP modeling the non-standard inference need at hand. It is easy to notice that, therefore, the whole approach relies on the logic rules formalizing structural subsumption, which is at the basis of each formula to be solved.

The crucial role of subsumption affects the system architecture in Figure 1, whose main components are described below:

- **Subs** is the central component, which implements a recursive algorithm solving subsumption between concept descriptions; such a module is designed to provide one interface for each DL adopted to model the problem: the current prototype allows for solving subsumption in \mathcal{EL} and \mathcal{ALN} .
- **Problems** is the component implementing OSP solving algorithms: the current prototype allows for solving Least Common Subsumer (**ics**), Concept Difference (**diff**) and Concept Abduction (**abd**), but **Problems** may be extended to include further services. It is noteworthy how, depending on the DL adopted to model the problem, a different subsumption interface, either **subs_EL** or **subs_ALN**, is invoked.
- **Support Modules** includes clauses supporting the performance of subsumption and inferences included in Problems, but related to sorts of information processing outside the core solving algorithms, such as \mathcal{ALN} concept normalization in Concept Centered Normal Form and special purpose lists manipulation.

² See <http://dl.dropbox.com/u/28260263/DL2012exe.rar> for an executable version of the prototype.

4 Inverting Subsumption

In order to show the prototype implementing the solving strategy detailed so far, we refer to Least Common Subsumer computation, solved by the Prolog code fragment in the following, excerpted from **Problems** module.

```
1 :-use_module('subs_el').
2 :-use_module('subs_aln').
3 :-use_module('support_modules').
4 :-use_module('normalization').

5 problem(lcs,C,D,Result,DL):- lcs(C,D,Result,DL).
6 problem(abd,C,D,Result,DL):- abd(C,D,Result,DL).
7 problem(diff,C,D,Result,DL):- diff(C,D,Result,DL).

8 lcs(C1, C2, LN, DL) :-
9     manage_concept(C1, C1N, DL),
10    manage_concept(C2, C2N, DL),
11    find_lcs(C1N, C2N, [top], L, DL),
12    normalization_top(L, LN).

13 find_lcs(C1, C2, L1, L3, DL) :-
14    decorate(L1, L2),
15    better_lcs(C1, C2, L1, L2, DL), !,
16    find_lcs(C1, C2, L2, L3, DL).
17 find_lcs(C1, C2, L1, L1, _).

18 decorate(C, C0) :- list(C, CL), select(some(R, D), CL, Rest),
19    decorate(D, DL), append(Rest, [some(R, DL)], C0).
20 decorate(C, C0) :- list(C, CL), append(CL, [X0], C0).

21 better_lcs(C1, C2, L1, L2, el) :-
22    subs_el(C1, L2),
23    subs_el(C2, L2),
24    not(subs_el(L1, L2)).

25 better_lcs(C1, C2, L1, L2, aln) :-
26    computeMaxAtLeast(C1, Max3),
27    computeMaxAtLeast(C2, Max4),
28    MaxL is max(Max3, Max4),
29    computeMaxAtMost(C1, Max1),
30    computeMaxAtMost(C2, Max2),
31    MaxM is max(Max1, Max2),
32    subs_aln(C1, L2, MaxL, MaxM),
33    subs_aln(C2, L2, MaxL, MaxM),
34    not(subs_aln(L1, L2, MaxL, MaxM)).
```

We shortly recall that the shared strategy we proposed relies on the solution of an **Optimal Solution Problem** in which we search for solutions which are optimal w.r.t. a given preorder, by incrementally trying to *find* solutions which are *better* than the one at hand, till a best one is reached.

In order to compute the Least Common Subsumer LN of two concepts, C_1 and C_2 in a DL (see line 8), we need to incrementally construct a concept which subsumes both C_1 and C_2 , and is optimal w.r.t. subsumption minimality (in fact, LN must be the most specific common subsumer of C_1 and C_2). To this aim, we start considering the trivial, subsumption maximal, solution, $L_1 = \top$ (line 11) and recursively try to *find* (lines 13–16) *better* common subsumers L_2 (line 15), by solving the system reported hereafter: $\{C_1 \sqsubseteq L_2; C_2 \sqsubseteq L_2; L_1 \not\sqsubseteq L_2\}$ (lines 22–24 or 32–34, depending on the adopted \mathcal{DL}). When no common subsumer L_n such that $L_{n-1} \not\sqsubseteq L_n$ exists, L_{n-1} is returned as best (Least) Common Subsumer (line 17).

The incremental construction of candidate better common subsumers L_2 exploits a predicate, namely *decorate*, which makes the common subsumer at hand L_1 more specific by appending fresh variables to it at every nesting level (lines 18–20). We notice that, even though different clauses are needed to check if L_2 is *better* than L_1 in \mathcal{EL} (lines 21–24) and \mathcal{ALN} (lines 25–34), such a distinction is only due to efficiency reasons: *subs_aln* needs two parameters more than *subs_el* and the adoption of a logic-independent unique *better_lcs* would force *subs_el* to work less efficiently with such two parameters, even though instantiated to anonymous variables. By the way, the reader can notice that the solving strategy underlying *better* is shared by both characterizations.

We observe also that all predicates invoked but not listed in the previously reported excerpt belong to one of the imported modules. In particular, *subs_el* and *sub_aln* modules provide the related logic-dependent subsumption programs, listed in Section 4.1. The other imported modules, *i.e.*, *support_modules* and *normalization*, include clauses crucial for the problem solution, but outside the core solving algorithms.

Among the others, we underline the role of the logic-dependent predicate *manage_concept* (see lines 9–10), which manipulates input concepts to make them ready for subsumption in the adopted DL : in the case of \mathcal{EL} , simple list manipulation operations are performed, while in the case of \mathcal{ALN} , such a predicate starts the process of normalization of input concepts: concepts are reduced in CCNF and both possible clashes and inherent subsumption relationships between number restrictions are identified.

4.1 Subsumption

Both in \mathcal{EL} and in \mathcal{ALN} , the subsumption algorithm takes as input concept descriptions written as conjunctions, formalized as Prolog lists. We recall that in \mathcal{ALN} such Prolog lists result from a pre-processing step of problem inputs: before checking for subsumption, concepts are manipulated to identify and manage possible clashes, number restrictions relationships and reduction in CCNF.

Given two concept descriptions C_1 and C_2 in a DL \mathcal{DL} , in order to prove whether C_2 subsumes C_1 (formally $C_1 \sqsubseteq C_2$), the algorithm recursively searches, for each member of the list related to C_2 , at least one subsumed member in the list represent-

ing $C1$. In other words, the whole subsumption check mechanism reverts to a one-one comparison between list members (or, more appropriately, conjuncts).

With ground lists, the proposed subsumption predicate just returns boolean answers showing check results. Nevertheless, we notice that conjuncts in input concept descriptions may also include concept variables: when lists are not ground, subsumption is inverted to exhibit possible variables substitutions making subsumption between list members true. The mechanism exploits Prolog built-in unification.

As hinted before, the overall mechanism solving subsumption is shared by both implementations and is built on one-to-one comparison of list members, either ground or variables.

Clauses comparing single list members exploit syntactical features of the DL at hand to either check subsumption between ground elements or unify variables to values making subsumption true. In the following, the Prolog code for such clauses in both implementations is provided.

Subsumption in \mathcal{EL}

```

1 subsomeone(A, A, BL, BLF):-
2     literal(A),
3     not(member(A, BL)),
4     append([A], BL, BLF).
5 subsomeone(some(R,C1), some(R, C2N), BL, BLF):-
6     subs_el(C1, C2),
7     normalization_top(C2, C2N),
8     not(subs_el(BL,some(R,C2N))),
9     append([some(R,C2N)],BL,BLF) .
10 subsomeone(Any, top, [], [top]).

11 subsomeoneground(A, A):- literal(A).
12 subsomeoneground(some(R,C1), some(R, C2)):-
13     subs_el(C1, C2).
14 subsomeoneground(_, top).

```

Subsumption in \mathcal{ALN}

```

1 subsomeone(bottom, _, _, _).
2 subsomeone(_, top, _, _).
3 subsomeone(A, A, _, _):- literal(A).
4 subsomeone(atleast(N,R), atleast(M, R), _, _):-
5     integer(N),
6     integer(M),
7     >=(N, M).
8 subsomeone(atleast(N,R), atleast(M, R),_, _):-
9     var(M),
10     geqpositive(N,M).
11 subsomeone(atleast(N,R), atleast(M, R), MaxL, _):-

```

```

12     var (N) ,
13     integer (M) ,
14     integer (MaxL) ,
15     leqBounded (M,N,MaxL) .
16 subsoneone (atmost (N,R) , atmost (M, R) ,_, _ ) :-
17     integer (N) ,
18     integer (M) ,
19     ! ,
20     =< (N,M) .
21 subsoneone (atmost (N,R) , atmost (M, R) ,_, MaxM) :-
22     var (M) ,
23     integer (MaxM) ,
24     leqBounded (N, M, MaxM) .
25 subsoneone (atmost (N,R) , atmost (M, R) ,_, _ ) :-
26     var (N) ,
27     geqpositive (M,N) .
28 subsoneone (_, all (R, top) , MaxL, MaxM) :-
29     nonvar (R) .
30 subsoneone (all (R,C1) , all (R, C2) , MaxL, MaxM) :-
31     subsoneone (C1, C2, MaxL, MaxM) .
32 subsoneone (atmost (0, R) , all (R, C) , _, _) .

```

5 Querying the Prototype

In order to show our prototype working mode, we refer to the examples in the following, related to the three computational problems and the two DLs investigated in the paper:

1. $L = LCS(C_1, C_2)$, $DL = \mathcal{EL}$
 $C_1 = \exists R.(A \sqcap B) \sqcap \exists R.(C \sqcap D)$;
 $C_2 = \exists R.(A \sqcap C) \sqcap \exists R.(B \sqcap D)$
2. $L = LCS(C_1, C_2)$, $DL = \mathcal{ALN}$
 $C_1 = (\geq 3 G) \sqcap (\leq 7 S) \sqcap \forall R.(\leq 2 M)$;
 $C_2 = (\geq 4 G) \sqcap (\leq 3 S) \sqcap \forall R.U$
3. $L = DIFF(C_1, C_2)$, $DL = \mathcal{EL}$
 $C_1 = A \sqcap B \sqcap \exists R.(C \sqcap D \sqcap \exists S.(H \sqcap J))$;
 $C_2 = A \sqcap B \sqcap \exists R.(\exists S.H)$
4. $L = DIFF(C_1, C_2)$, $DL = \mathcal{ALN}$
 $C_1 = A \sqcap \forall R.(B \sqcap (\leq 4 S)) \sqcap (\leq 0 T)$;
 $C_2 = A \sqcap \forall R.(\leq 4 S) \sqcap \forall T.(D \sqcap \forall U.E \sqcap (\geq 2 V))$
5. $L = ABD(C_1, C_2)$, $DL = \mathcal{EL}$
 $C_1 = \exists R.(\exists S.H)$;
 $C_2 = A \sqcap B \sqcap \exists R.(C \sqcap D \sqcap \exists S.(H \sqcap J))$
6. $L = ABD(C_1, C_2)$, $DL = \mathcal{ALN}$
 $C_1 = (\geq 2 R) \sqcap \forall R.\neg A \sqcap B, \sqcap C$;
 $C_2 = B \sqcap (\geq 3 R)$

Table 1 shows the Prolog formalization and the results for the queries corresponding to the problems above. We note that, when problems admit multiple solutions — as it is in

Table 1. Prolog Queries

Query	Formalization	Result
1	<i>problem(lcs, [some(r,[a,b]), some(r,[c,d])], [some(r,[a,c]), some(r,[b,d])], L, el)</i>	$L = [some(r,[a]), some(r,[b]), some(r,[c]), some(r,[d])]$
2	<i>problem(lcs, [atleast(3,g), atmost(7,s), all(r,atmost(2,m))], [atleast(4,g), atmost(3,s), all(r,u)], L, aln)</i>	$L = [atleast(3, g), atmost(7, s)]$
3	<i>problem(diff, [a,b, some(r, [c,d, some(s,[h,j])]), [a,b, some(r,[some(s, [h])])], L, el)</i>	$L = [some(r, [c, d, some(s, [h, j])])]$
4	<i>problem(diff, [a, all(r, [b, atmost(4, s)]), atmost(0, t)], [a, all(r, atmost(4, s)), all(t, [d,all(u, e), atleast(2,v)])], L, aln)</i>	$L = [atmost(0, t), all(r, b)]$
5	<i>problem(abd, [some(r,[some(s, [h])])], [a,b, some(r,[c,d, some(s,[h,j])])], L, el)</i>	$L = [a,b, some(r, [c, d, some(s, [h, j])])]$
6	<i>problem(abd, [atleast(2,r),all(r,neg(a)), b, c],[b, atleast(3, r)],L, aln)</i>	$L = [atleast(3, r)]$

Concept Abduction and Concept Difference—the system stops searching for solutions when the first one is retrieved. As pointed out since the introduction, our prototype is still inefficient at this stage: all results in Table 1 need a few seconds to be returned, and Query 4, which is the most complex one, asks for about 10 seconds.³

6 Discussion and Future Work

Motivated by the need to unify as much as possible the process of solving non-standard reasoning problems, we proposed a general framework dealing with several inferences according to a logic-independent strategy, to be further specialized to cope with the DL adopted to model the problem at hand.

The paper presents a modular Logic Programming prototype system demonstrating the feasibility of the proposed strategy for Least Common Subsumer, Concept Difference and Concept Abduction computation in \mathcal{EL} and \mathcal{ALN} .

The extension of the approach to different DL sublanguages, and the implementation, for each investigated DL, of further non-standard reasoning services in the prototype is part of our future work, together with the improvement of system efficiency.

Of course, the approach presented in this paper has some theoretical limitations. Namely, the use of structural subsumption limits this approach to DLs for which structural subsumption is complete. For more expressive DLs, the fixpoint mechanism could

³ Using an Intel(R) Core(TM) i5 CPU 2.40 GHz with 4.00 GB RAM.

still be exploited, but using some higher-order tableaux methods that are still to be defined and whose correctness and termination should be proved.

References

1. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Proc. of IJCAI 2003. pp. 319–324 (2003)
2. Baader, F., Küsters, R., Borgida, A., McGuinness, D.L.: Matching in description logics. *J. of Log. and Comp.* 9(3), 411–447 (1999)
3. Baader, F., Küsters, R., Molitor, R.: Rewriting concepts using terminologies. In: Proc. of KR 2000. pp. 297–308 (2000)
4. Baader, F., Morawska, B.: Unification in the description logic \mathcal{EL} . *Logical Methods in Computer Science* 6(3) (2010)
5. Baader, F., Narendran, P.: Unification of concept terms in description logics. *J. of Symbolic Computation* 31, 277–305 (2001)
6. Baader, F., Sertkaya, B.: Usability issues in description logic knowledge base completion. In: ICFA-2009. pp. 1–21 (2009)
7. Baader, F., Sertkaya, B., Turhan, A.Y.: Computing the least common subsumer w.r.t. a background terminology. *J. of Appl. Log.* 5(3), 392–420 (2007)
8. Borgida, A., Walsh, T., Hirsh, H.: Towards measuring similarity in description logics. In: Proc. of DL 2005 (2005)
9. Cohen, W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: Rosenbloom, P., Szolovits, P. (eds.) Proc. of AAAI’92. pp. 754–761. AAAI Press, Menlo Park, California (1992)
10. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: A unified framework for non-standard reasoning services in description logics. In: Proc. of ECAI 2010. pp. 479–484. Lisbon, Portugal, August 16-20 (2010)
11. Di Noia, T., Di Sciascio, E., Donini, F.M.: Semantic matchmaking as non-monotonic reasoning: A description logic approach. *J. of Artificial Intell. Res.* 29, 269–307 (2007)
12. Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M.: Abductive matchmaking using description logics. In: Proc. of IJCAI 2003. pp. 337–342 (2003)
13. Kazakov, Y., Krötzsch, M., Simancik, F.: Concurrent classification of \mathcal{EL} ontologies. In: International Semantic Web Conference (1). pp. 305–320 (2011)
14. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proc. of IJCAI 2009. pp. 830–835. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2009), <http://dl.acm.org/citation.cfm?id=1661445.1661577>
15. Lecue, F., Kotoulas, S., Aonghusa, P.M.: Capturing the pulse of cities: A robust stream data reasoning approach. Position paper, IBM Research, Smarter Cities Technology Centre, Dublin, Ireland (2011), wiki.planet-data.eu
16. McGuinness, D.L., Borgida, A.: Explaining subsumption in description logics. In: Proc. of IJCAI’95. pp. 816–821 (1995)
17. Nikitina, N.: Uniform interpolation in general \mathcal{EL} terminologies. Techreport, Institut AIFB, KIT, Karlsruhe (Mai 2011)
18. Schlobach, S.: Explaining subsumption by optimal interpolation. In: Proc. of JELIA’2004. pp. 413–425 (2004)
19. Teege, G.: Making the difference: A subtraction operation for description logics. In: Proc. of KR’94. pp. 540–550 (1994)

An EXPSPACE Tableau-based Algorithm for \mathcal{SHOIQ}

Chan Le Duc¹, Myriam Lamolle¹, and Olivier Curé²

¹ LIASD Université Paris 8 - IUT de Montreuil, France
{chan.leduc, myriam.lamolle}@iut.univ-paris8.fr
² LIGM Université Paris-Est, France
ocure@univ-mlv.fr

Abstract. In this paper, we propose an EXPSPACE tableau-based algorithm for \mathcal{SHOIQ} . The construction of this algorithm is founded on the standard tableau-based method for \mathcal{SHOIQ} and the technique used for designing a NEXPTIME algorithm for the two-variable fragment of first-order logic with counting quantifiers \mathcal{C}^2 .

1 Introduction

The ontology language OWL-DL [6] is widely used to formalize semantic resources on the Semantic Web. This language is mainly based on the description logic \mathcal{SHOIQ} which is known to be decidable [9]. There have been several works on the consistency problem of a \mathcal{SHOIQ} knowledge base. These works have not only shown decidability and complexity of the problem but also led to develop and implement efficient systems for reasoning on OWL-based ontologies. Tobies [9] has shown that the consistency problem of a \mathcal{SHOIQ} knowledge base is NEXPTIME-complete. Horrocks *et al.* [2] have proposed a tableau-based algorithm that has been exploited to implement reasoners such as Pellet [8], which inherit from the success of early Description Logic reasoners such as FaCT [1].

It has been shown that when nominals are added to DLs the consistency problem is harder. In fact, the complexity jumps from EXPTIME-complete for \mathcal{SHIQ} to NEXPTIME-complete for \mathcal{SHOIQ} [9]. Kazakov *et al.* [4] have indicated that when nominals are allowed in \mathcal{SHIQ} , the resolution-based approach yields a triple exponential decision procedure for the consistency problem. The authors have also identified that the interaction between nominals, inverse roles and number restrictions makes termination more difficult to be achieved, and thus, is responsible for this hardness.

Our approach is inspired from the technique that was employed by Pratt-Hartmann [7] to construct a NEXPTIME algorithm for the logic \mathcal{C}^2 that almost includes \mathcal{SHOIQ} . Unlike the existing tableau-based algorithms, this technique does not explicitly build a graph for representing a model but it builds a structure, called a *frame*, from *star-types* each of which represents a set of individuals. Pratt-Hartmann [7] shows that a model of a \mathcal{C}^2 knowledge base can be constructed from a frame tiled by *well selected* star-types.

The present paper is structured as follows. In the next section, we describe the logic \mathcal{SHOIQ} and the consistency problem for a \mathcal{SHOIQ} knowledge base. Section 3 describes a 2EXPSPACE tableau-based algorithm for checking consistency of a \mathcal{SHOIQ}

knowledge base. An advantage of this algorithm is that a tree-like structure can be maintained to obtain termination. Section 4 transfers results from \mathcal{C}^2 [7] to \mathcal{SHOIQ} , and presents an EXPSPACE tableau-based algorithm for \mathcal{SHOIQ} . Finally, we discuss the results and future work. For the lack of place, we refer the reader to [5] for examples and full proofs.

2 The Description Logic \mathcal{SHOIQ}

In this section, we present the syntax and the semantics of \mathcal{SHOIQ} . We start by defining a role hierarchy and its semantics.

Definition 1 (role hierarchy). Let \mathbf{R} be a non-empty set of role names and $\mathbf{R}_+ \subseteq \mathbf{R}$ be a set of transitive role names. We use $\mathbf{R}_1 = \{P^- \mid P \in \mathbf{R}\}$ to denote a set of inverse roles. Each element of $\mathbf{R} \cup \mathbf{R}_1$ is called a \mathcal{SHOIQ} -role. We define $R^\ominus := R^-$ if $R \in \mathbf{R}$, and $R^\ominus := R$ if $R \in \mathbf{R}_1$. A role hierarchy \mathcal{R} is a finite set of role inclusion axioms $R \sqsubseteq S$ where R and S are two \mathcal{SHOIQ} -roles. A relation \sqsubseteq is defined as the transitive-reflexive closure of \sqsubseteq on $\mathcal{R} \cup \{R^\ominus \sqsubseteq S^\ominus \mid R \sqsubseteq S \in \mathcal{R}\}$. We define a function $\text{Trans}(R)$ which returns true iff there is some $Q \in \mathbf{R}_+ \cup \{P^\ominus \mid P \in \mathbf{R}_+\}$ such that $Q \sqsubseteq R$. A role R is called simple w.r.t. \mathcal{R} if $\text{Trans}(Q) = \text{false}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ (domain) and a function $\cdot^{\mathcal{I}}$ which maps each role name to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that $R^{-\mathcal{I}} = \{\langle x, y \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle y, x \rangle \in R^{\mathcal{I}}\}$ for all $R \in \mathbf{R}$, and $\langle x, z \rangle \in S^{\mathcal{I}}, \langle z, y \rangle \in S^{\mathcal{I}}$ implies $\langle x, y \rangle \in S^{\mathcal{I}}$ for each $S \in \mathbf{R}_+$. An interpretation \mathcal{I} satisfies a role hierarchy \mathcal{R} if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \sqsubseteq S \in \mathcal{R}$. Such an interpretation is called a model of \mathcal{R} , denoted by $\mathcal{I} \models \mathcal{R}$.

Definition 2 (terminology). Let \mathbf{C} be a non-empty set of concept names with a non-empty subset $\mathbf{C}_o \subseteq \mathbf{C}$ of nominals. The set of \mathcal{SHOIQ} -concepts is inductively defined as the smallest set containing all C in \mathbf{C} , \top , $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists R.C$, $\forall R.C$, $(\leq n S.C)$ and $(\geq n S.C)$ where n is a positive integer, C and D are \mathcal{SHOIQ} -concepts, R is an \mathcal{SHOIQ} -role and S is a simple role w.r.t. a role hierarchy. We denote \perp for $\neg \top$. The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps each concept name to a subset of $\Delta^{\mathcal{I}}$ such that $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $\text{card}\{o^{\mathcal{I}}\} = 1$ for all $o \in \mathbf{C}_o$, $(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, \langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$, $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}, \langle x, y \rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$, $(\geq n S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{card}\{y \in C^{\mathcal{I}} \mid \langle x, y \rangle \in S^{\mathcal{I}}\} \geq n\}$, $(\leq n S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{card}\{y \in C^{\mathcal{I}} \mid \langle x, y \rangle \in S^{\mathcal{I}}\} \leq n\}$ where $\text{card}\{S\}$ is denoted for the cardinality of a set S .

* $C \sqsubseteq D$ is called a general concept inclusion (GCI) where C, D are \mathcal{SHOIQ} -concepts (possibly complex), and a finite set of GCIs is called a terminology \mathcal{T} .

* An interpretation \mathcal{I} satisfies a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and \mathcal{I} satisfies a terminology \mathcal{T} if \mathcal{I} satisfies each GCI in \mathcal{T} . Such an interpretation is called a model of \mathcal{T} , denoted by $\mathcal{I} \models \mathcal{T}$.

Definition 3 (knowledge base). A pair $(\mathcal{T}, \mathcal{R})$ is called a \mathcal{SHOIQ} knowledge base where \mathcal{R} is a \mathcal{SHOIQ} role hierarchy and \mathcal{T} is a \mathcal{SHOIQ} terminology. A knowledge

base $(\mathcal{T}, \mathcal{R})$ is said to be consistent if there is a model \mathcal{I} of both \mathcal{T} and \mathcal{R} , i.e., $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{R}$. A concept C is called satisfiable w.r.t. $(\mathcal{T}, \mathcal{R})$ iff there is some interpretation \mathcal{I} such that $\mathcal{I} \models \mathcal{R}$, $\mathcal{I} \models \mathcal{T}$ and $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a model of C w.r.t. $(\mathcal{T}, \mathcal{R})$. A concept D subsumes a concept C w.r.t. $(\mathcal{T}, \mathcal{R})$, denoted by $C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in each model \mathcal{I} of $(\mathcal{T}, \mathcal{R})$.

Thanks to the reductions between unsatisfiability, subsumption of concepts and knowledge base consistency, it suffices to study knowledge base consistency.

For the ease of construction, we assume all concepts to be in *negation normal form* (NNF), i.e., negation occurs only in front of concept names. Any *SHOIQ*-concept can be transformed to an equivalent one in NNF by using DeMorgan's laws and some equivalences as presented in [3]. For a concept C , we denote the nnf of C by $\text{nnf}(C)$ and the nnf of $\neg C$ by \dot{C} . Let D be an *SHOIQ*-concept in NNF. We define $\text{cl}(D)$ to be the smallest set that contains all sub-concepts of D including D . For a knowledge base $(\mathcal{T}, \mathcal{R})$, we can define a set $\text{cl}(\mathcal{T}, \mathcal{R})$. For the sake of brevity, we refer the reader to [2] for a more complete definition.

To prove soundness and completeness of our algorithms, we need a tableau structure that represents a model of a *SHOIQ* knowledge base. Regarding the definition of tableaux for *SHOIQ* presented in [2], we add a new property that imposes an exact number of *S*-neighbour individuals t of s if $(\leq nS.C) \in \mathcal{L}(s)$. This property makes explicit non-determinism implied from the semantics of $(\leq nS.C)$ and requires an extra expansion rule for the tableau-based algorithm.

3 A 2EXPSPACE Decision Procedure for *SHOIQ*

In this section, we introduce a structure, called *SHOIQ*-forest. We will show that such a forest is sufficient to represent a model of a *SHOIQ*-knowledge base.

Definition 4 (tree). Let $(\mathcal{T}, \mathcal{R})$ be a *SHOIQ* knowledge base. For each $o \in \mathbf{C}_o$, a *SHOIQ*-tree for $(\mathcal{T}, \mathcal{R})$, denoted by $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \hat{x}_o, \neq_o)$, is defined as follows:

- * V_o is a set of nodes containing a root node $\hat{x}_o \in V_o$. Each node $x \in V_o$ is labelled with a function \mathcal{L}_o such that $\mathcal{L}_o(x) \subseteq \text{cl}(\mathcal{T}, \mathcal{R})$ and $o \in \mathcal{L}_o(\hat{x}_o)$. A node $x \in V_o$ is called nominal if $o' \in \mathcal{L}_o(x)$ for some $o' \in \mathbf{C}_o$. In addition, the inequality relation \neq_o is a symmetric binary relation over V_o .

- * E_o is a set of edges. Each edge $\langle x, y \rangle \in E_o$ is labelled with a function \mathcal{L}_o such that $\mathcal{L}_o(\langle x, y \rangle) \subseteq \mathbf{R}_{(\mathcal{T}, \mathcal{R})}$. If $\langle x, y \rangle \in E_o$ then y is called a successor of x , denoted by $y \in \text{succ}^1(x)$, or x is called the predecessor of y , denoted by $x = \text{pred}^1(y)$. In this case, we say that x is a neighbour of y or y is a neighbour of x . If $z \in \text{succ}^n(x)$ (resp. $z = \text{pred}^n(x)$) and y is a successor of z (resp. y is the predecessor of z) then $y \in \text{succ}^{(n+1)}(x)$ (resp. $y = \text{pred}^{(n+1)}(x)$) for all $n \geq 0$ where $\text{succ}^0(x) = \{x\}$ and $\text{pred}^0(x) = x$. A node y is called a descendant of x if $y \in \text{succ}^n(x)$ for some $n > 0$. A node y is called an ancestor of x if $y = \text{pred}^n(x)$ for some $n > 0$. To ensure that \mathbf{T}_o is a tree, it is required that (i) x is a descendant of \hat{x}_o for all $x \in V_o$ with $x \neq \hat{x}_o$, and (ii) each node $x \in V_o$ with $x \neq \hat{x}_o$ has a unique predecessor. A node y is called an *R*-successor of x , denoted by $y \in \text{succ}_R^1(x)$ (resp. y is called the *R*-predecessor

of x , denoted by $y = \text{pred}_R^1(x)$ if there is some role R' such that $R' \in \mathcal{L}_o(\langle x, y \rangle)$ (resp. $R' \in \mathcal{L}_o(\langle y, x \rangle)$) and $R' \not\sqsubseteq R$. A node y is called a R -neighbour of x if y is either a R -successor or R -predecessor of x . If z is an R -successor of y (resp. z is the R -predecessor of y) and $y \in \text{succ}_R^n(x)$ (resp. $y = \text{pred}_R^n(x)$) then $z \in \text{succ}_R^{(n+1)}(x)$ (resp. $z = \text{pred}_R^{(n+1)}(x)$) for $n \geq 0$ with $\text{succ}_R^0(x) = \{x\}$ and $x = \text{pred}_R^0(x)$.

* For a node x , a role S and $o \in \mathbf{C}_o$, we define the set $S^{\mathbf{T}_o}(x, C)$ of x 's S -neighbours as follows: $S^{\mathbf{T}_o}(x, C) = \{y \in V_o \mid y \text{ is a } S\text{-neighbour of } x \text{ and } C \in \mathcal{L}_o(x)\}$.

* A node x is called iterated by y w.r.t. a node x_o if x has no nominal ancestor except for \hat{x}_o and there are integers $n, m > 0$ and nodes x', y' such that : (i) $x_o = \text{pred}^n(y)$, $y = \text{pred}^m(x)$, (ii) $x' = \text{pred}^1(x)$, $y' = \text{pred}^1(y)$, (iii) $\mathcal{L}_o(x) = \mathcal{L}_o(y)$, $\mathcal{L}_o(x') = \mathcal{L}_o(y')$, (iv) $\mathcal{L}_o(\langle x', x \rangle) = \mathcal{L}_o(\langle y', y \rangle)$, and (v) if there are z, z' and $i > 0$ such that $z' = \text{pred}^1(z)$, $\text{pred}^i(z') = x_o$, $\mathcal{L}_o(z) = \mathcal{L}_o(y)$, $\mathcal{L}_o(z') = \mathcal{L}_o(y')$ and $\mathcal{L}_o(\langle z', z \rangle) = \mathcal{L}_o(\langle y', y \rangle)$ then $i \geq n$.

A node x is called 1-iterated by y if x is iterated by y w.r.t. \hat{x}_o . A node x is called blocked by y , denoted by $y = \mathbf{b}(x)$, if x is iterated by y w.r.t. a 1-iterated node x_o .

* In the following, we often use $\mathcal{L}(x)$, $\mathcal{L}(\langle x, y \rangle)$, $S^{\mathbf{T}}(x, C)$ and \neq instead of $\mathcal{L}_o(x)$, $\mathcal{L}_o(\langle x, y \rangle)$, $S^{\mathbf{T}_o}(x, C)$ and \neq_o , respectively. This does not cause any confusion since $V_o \cap V_{o'} = \emptyset$ and $E_o \cap E_{o'} = \emptyset$ if $o \neq o'$. In addition, $x \neq_o y$ is never defined for $x \in V_o$ and $y \in V_{o'}$ with $o \neq o'$.

We remark that the definition of 1-iterated nodes in Definition 4 for \mathcal{SHOIQ} -trees is very similar to the standard definition of blocked nodes for \mathcal{SHIQ} completion trees (see [3]). Moreover, if we consider the sub-tree rooted at a 1-iterated node as a \mathcal{SHIQ} completion tree then blocked nodes according to Definition 4 are also blocked nodes according to the standard definition for this \mathcal{SHIQ} completion tree.

A \mathcal{SHOIQ} -tree consists of two layers : the first layer is formed of nodes from the root to 1-iterated nodes or nominal nodes, and the second layer consists of nodes from each 1-iterated node to blocked or nominal nodes. In addition, each node x in the layer 2 has a unique 1-iterated node, denoted $\hat{\mathbf{b}}(x)$, such that $\hat{\mathbf{b}}(x)$ is an ancestor of x .

Definition 5 (forest). Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. A \mathcal{SHOIQ} -forest for $(\mathcal{T}, \mathcal{R})$ is a pair $\mathbf{G} = \langle \mathbf{T}, \varphi \rangle$, where $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ is a set of \mathcal{SHOIQ} -trees for $(\mathcal{T}, \mathcal{R})$ with $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \hat{x}_o, \neq_o)$, and φ is a partitioning function $\varphi : \mathcal{V} \rightarrow 2^{\mathcal{V}}$ with $\mathcal{V} = \bigcup_{o \in \mathbf{C}_o} V_o$. We denote $\mathcal{L}'(\langle x, y \rangle) = \mathcal{L}_o(\langle x, y \rangle)$ if $\langle x, y \rangle \in E_o$, and $\mathcal{L}'_o(\langle x, y \rangle) = \{S^{\ominus} \mid S \in \mathcal{L}_o(\langle y, x \rangle)\}$ if $\langle y, x \rangle \in E_o$ for some $o \in \mathbf{C}_o$. The partitioning function φ satisfies the following conditions:

1. For each $x \in \mathcal{V}$, $\varphi(x)$ is the partition of x with $x \in \varphi(x)$. There are $x_0, \dots, x_n \in \mathcal{V}$ such that $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ with $0 \leq i < j \leq n$ and $\bigcup_{0 \leq i \leq n} \varphi(x_i) = \mathcal{V}$;
2. For all $x, x' \in \mathcal{V}$, if $x' \in \varphi(x)$ then $\varphi(x) = \varphi(x')$ and $\mathcal{L}(x) = \mathcal{L}(x')$. We denote $\Lambda(\varphi(x)) = \mathcal{L}(x)$. In addition, an inequality relation over partitions can be described as follows : for $x, x' \in \mathcal{V}$ we define $\varphi(x) \neq \varphi(x')$ if there are two nodes $y \in \varphi(x)$ and $y' \in \varphi(x')$ such that $y \neq_o y'$ for some $o \in \mathbf{C}_o$;
3. For all $\varphi(x)$ and $\varphi(x')$, if there are two edges $\langle y, y' \rangle \in E_o$ and $\langle w, w' \rangle \in E_{o'}$ with $o, o' \in \mathbf{C}_o$ such that $y, w \in \varphi(x)$, $y', w' \in \varphi(x')$, $\mathcal{L}'(\langle y, y' \rangle) \neq \emptyset$, $\mathcal{L}'(\langle w, w' \rangle) \neq \emptyset$ then $\mathcal{L}'(\langle y, y' \rangle) = \mathcal{L}'(\langle w, w' \rangle)$.

We define a function $\Lambda(\langle \cdot, \cdot \rangle)$ for labelling edges ended by two partitions as follows: $\Lambda(\langle \varphi(x), \varphi(x') \rangle) = \mathcal{L}'(\langle z, z' \rangle)$ where $z \in \varphi(x)$, $z' \in \varphi(x')$, $\mathcal{L}'(\langle z, z' \rangle) \neq \emptyset$, and $\{\langle z, z' \rangle, \langle z', z \rangle\} \cap E_{o'} \neq \emptyset$ for some $o' \in \mathbf{C}_o$. We say $\varphi(x')$ is a S -neighbour partition of $\varphi(x)$ if $S \in \Lambda(\langle \varphi(x), \varphi(x') \rangle)$.

4. For all $x, x' \in \mathcal{V}$, if $o \in \mathcal{L}(x) \cap \mathcal{L}(x')$ for some $o \in \mathbf{C}_o$ and $\varphi(x) \neq \varphi(x')$ does not hold then $\varphi(x) = \varphi(x')$;
5. If $(\leq nR.C) \in \Lambda(\varphi(x))$ for some $x \in \mathcal{V}$ and there exist $(n+1)$ nodes $x_0, \dots, x_n \in \mathcal{V}$ such that (i) $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ for all $0 \leq i < j \leq n$, and (ii) $C \in \Lambda(\varphi(x_i))$, $R \in \Lambda(\langle \varphi(x), \varphi(x_i) \rangle)$ for all $i \in \{0, \dots, n\}$, then $\varphi(x_i) \neq \varphi(x_m)$ for all $0 \leq i < m \leq n$; and
6. If $(\geq nR.C) \in \Lambda(\varphi(x))$ for some $x \in \mathcal{V}$ then $\varphi(x)$ has n R -neighbour partitions $\varphi(x_1), \dots, \varphi(x_n)$ such that $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ and $C \in \Lambda(\varphi(x_i))$ for all $1 \leq i < j \leq n$.

* **Clashes:** \mathbf{T} is said to contain a clash if one of the following conditions holds:

1. There is some node $x \in \mathcal{V}$ such that $\{A, \dot{\neg}A\} \subseteq \Lambda(\varphi(x))$ for some concept name $A \in \mathbf{C}$;
2. There are nodes $x, y \in \mathcal{V}$ such that $\varphi(x) \neq \varphi(y)$ and $o \in \Lambda(\varphi(x)) \cap \Lambda(\varphi(y))$ for some $o \in \mathbf{C}_o$;
3. There is a node $x \in \mathcal{V}$ with $(\leq nR.C) \in \Lambda(\varphi(x))$ and there are $(n+1)$ nodes $x_0, \dots, x_n \in \mathcal{V}$ such that $\varphi(x_i) \cap \varphi(x_j) = \emptyset$, $\varphi(x_i) \neq \varphi(x_j)$ with $0 \leq i < j \leq n$, and $C \in \Lambda(\varphi(x_i))$, $R \in \Lambda(\langle \varphi(x), \varphi(x_i) \rangle)$ for $i \in \{0, \dots, n\}$.

We now describe the tableau-based algorithm whose goal is to construct from a knowledge base $(\mathcal{T}, \mathcal{R})$ a \mathcal{SHOIQ} -forest $\mathbf{G} = \langle \mathbf{T}, \varphi \rangle$. To do this, the algorithm applies expansion rules (as described in Fig. 1 and Fig. 2 in [5]), and terminates when none of the rules is applicable. The obtained \mathbf{G} is called *complete*, and if \mathbf{G} contains no clash then \mathbf{G} is called *clash-free*. In this case, we also say \mathbf{T}_o is complete and clash-free for all $\mathbf{T}_o \in \mathbf{T}$.

The expansion rules maintain the tree-like structure of \mathcal{SHOIQ} -forest and they are similar to those in [2] except that if a concept C is added to the label of a node x due to application of these rules then C is propagated to the label of each node $y \in \varphi(x)$. Moreover, all rules in Fig. 1 in [5] except for \exists - and \geq -rule update only the label of nodes or edges and do not change the partitioning function φ . In particular, when the \leq -rule is applied to a node x with two S -neighbours y, z of x , it must propagate the label of $\langle x, y \rangle$ to that of all $\langle x', z' \rangle$ (or $\langle z', x' \rangle$) where $x' \in \varphi(x)$ and $z' \in \varphi(z)$, and set the label of $\langle x, y \rangle$ to empty set. This may change φ only if $\varphi(y)$ is singleton. By a new rule, namely the \bowtie -rule, each node x containing a term $(\leq nS.C)$ has exactly m S -neighbours containing C with some $m \leq n$. As a result, this rule and \geq -rule ensure that if there are two nodes $y, y' \in \varphi(x)$ then y and y' have exactly m S -neighbours which contain C in their label. Finally, we can avoid infinite sequences of “merging-and-generating” without pruning nodes since all merges due to number restrictions or nominals are performed by updating the partitioning function. The following lemma establishes correctness and completeness of the algorithm.

Lemma 1. *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base.*

1. The tableau algorithm terminates and builds a \mathcal{SHOIQ} -forest whose the size is bounded by a doubly exponential function in the size of $(\mathcal{T}, \mathcal{R})$.
2. If the tableau algorithm yields a clash-free and complete \mathcal{SHOIQ} -forest for $(\mathcal{T}, \mathcal{R})$ then there is a tableau for $(\mathcal{T}, \mathcal{R})$.
3. If there is a tableau for $(\mathcal{T}, \mathcal{R})$ then the tableau algorithm yields a clash-free and complete \mathcal{SHOIQ} -forest for $(\mathcal{T}, \mathcal{R})$.

To prove soundness of the tableau algorithm, we can devise a model from a clash-free and complete \mathcal{SHOIQ} -forest by considering a partition as an individual and unravelling blocked nodes since we can show that each blocking node $b(x)$ has no “core path” from $b(x)$ to each nominal descendant y , i.e., there do not exist terms $(\leq m_i R_i, C_i) \in \text{pred}^i(y)$, roles $R_i \in \mathcal{L}(\langle \text{pred}^{i-1}(y), \text{pred}^i(y) \rangle)$ and concepts $C_i \in \mathcal{L}(\text{pred}^{i+1}(y))$ for $k < i \leq 0$, $b(x) = \text{pred}^k(y)$. The following theorem is a consequence of Lemma 1.

Theorem 1. *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. The tableau algorithm is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$ and it runs in 2NEXPTIME in the size of $(\mathcal{T}, \mathcal{R})$.*

4 An EXPSPACE Tableau-based Algorithm for \mathcal{SHOIQ}

This section starts by translating some results presented in [7] for \mathcal{C}^2 into those for \mathcal{SHOIQ} .

Definition 6 (star-type). *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. A star-type is a triplet $\sigma = \langle \lambda_\sigma, \bar{v}_\sigma, \bar{\mu}_\sigma \rangle$, where $\lambda_\sigma \in 2^{\text{cl}(\mathcal{T}, \mathcal{R})}$, $\bar{\mu}_\sigma = (\langle r_1, l_1 \rangle, \dots, \langle r_d, l_d \rangle)$ is a d -tuple over $2^{\mathbf{R}(\mathcal{T}, \mathcal{R})} \times 2^{\text{cl}(\mathcal{T}, \mathcal{R})}$, and $\bar{v}_\sigma = (\langle r', l' \rangle)$ with $\langle r', l' \rangle \in 2^{\mathbf{R}(\mathcal{T}, \mathcal{R})} \times 2^{\text{cl}(\mathcal{T}, \mathcal{R})}$. A pair $\langle r, l \rangle$ is a ray of σ if $\langle r, l \rangle$ is a component of $\bar{\mu}_\sigma$ or \bar{v}_σ . In particular, $\langle r, l \rangle$ is a predecessor ray if $(\langle r, l \rangle) = \bar{v}_\sigma$, and $\langle r, l \rangle$ is a successor ray if $\langle r, l \rangle$ is a component of $\bar{\mu}_\sigma$. We denote $\bar{\xi}_\sigma = (\langle r_1, l_1 \rangle, \dots, \langle r_d, l_d \rangle, \langle r_{d+1}, l_{d+1} \rangle)$ if $\bar{v}_\sigma = (\langle r', l' \rangle)$ where $r' = r_{d+1}$, $l' = l_{d+1}$, and $\bar{\xi}_\sigma = \bar{\mu}_\sigma$ if \bar{v}_σ is empty.*

- A ray $\langle r', l' \rangle$ of σ is primary w.r.t. a term $(\leq mR.C)$ if $(\leq mR.C) \in \lambda_\sigma$, $R \in r'$ and $C \in l'$. For a term $(\leq mR.C) \in \lambda_\sigma$, we denote $\mathcal{C}_{(\leq mR.C)}^\sigma$ for the set of all rays $\langle r', l' \rangle$ of σ such that $R \in r'$, $C \in l'$.
- A star-type σ is nominal if $o \in \lambda_\sigma$ for some $o \in \mathbf{C}_o$.
- A star-type σ is chromatic if there is a term $(\geq nS.D) \in \lambda_\sigma$ and σ has n rays $\langle r'_1, l'_1 \rangle, \dots, \langle r'_n, l'_n \rangle$ such that $S \in l'_i$, $D \in l'_i$ for all $1 \leq i \leq n$, and $l'_i \neq l'_j$ for all $0 \leq i < j \leq n$ with $l'_0 = \lambda_\sigma$.
- A star-type σ is homomorphic (resp. isomorphic) to a star-type σ' if $\lambda_\sigma = \lambda_{\sigma'}$, and for each term $(\leq mR.C) \in \lambda_\sigma$, there is an injection (resp. a bijection) $\pi : \mathcal{C}_{(\leq mR.C)}^\sigma \rightarrow \mathcal{C}_{(\leq mR.C)}^{\sigma'}$ such that $\pi(\langle r, l \rangle) = \langle r', l' \rangle$ implies $r' = r$ and $l' = l$.
- Two star-types σ, σ' are equivalent if $\lambda_\sigma = \lambda_{\sigma'}$, and there is a bijection π between $\bar{\xi}_\sigma$ and $\bar{\xi}_{\sigma'}$ such that $\pi(\langle r, l \rangle) = \langle r', l' \rangle$ implies $r' = r$ and $l' = l$.

We denote Σ for the set of all star-types for $(\mathcal{T}, \mathcal{R})$. ◁

In the context of a \mathcal{SHOIQ} -forest, we can think of a star-type σ as the set of nodes x such that $\mathcal{L}(x) = \lambda_\sigma$, and each ray $\langle r_i, l_i \rangle$ of σ corresponds to a neighbour x_i of x such that $\mathcal{L}'(\langle x, x_i \rangle) = r_i$ and $\mathcal{L}(x_i) = l_i$. In this case, we say that x satisfies σ .

Remark 1. The notion of chromaticity introduced in Definition 6 implies an inequality relation \neq over nodes. That stronger notion is needed to prevent “distinct” star-types from including nodes x, y which are neighbours or $x \neq y$. In order to make star-types chromatic, it is necessary to add to knowledge bases some new concepts and axioms as follows. Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. For each term $(\geq nS.D) \in \text{cl}(\mathcal{T}, \mathcal{R})$, we add to $\text{cl}(\mathcal{T}, \mathcal{R})$ n new concept names $C_{(\geq nS.D)}^0, \dots, C_{(\geq nS.D)}^n$, and to \mathcal{T} the following axioms: $C_{(\geq nS.D)}^i \sqcap C_{(\geq nS.D)}^j \sqsubseteq \perp$ for all $0 \leq i < j \leq n$. It is straightforward to prove that the terminology $(\mathcal{T}', \mathcal{R})$ is consistent iff $(\mathcal{T}, \mathcal{R})$ is consistent where \mathcal{T}' is obtained from \mathcal{T} by adding these new axioms. Thanks to these new concepts and axioms, the following definition points out how to build chromatic star-types.

Definition 7 (valid star-type). Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. Let σ be a star-type for $(\mathcal{T}, \mathcal{R})$ where $\sigma = \langle \lambda_\sigma, \bar{\nu}, \bar{\mu} \rangle$ with $\bar{\mu} = (\langle r_1, l_1 \rangle, \dots, \langle r_d, l_d \rangle)$ and $\lambda_\sigma = l_0$, $\bar{\nu} = \{\langle r_{d+1}, l_{d+1} \rangle\}$. σ is valid with an inequality relation \neq over C^σ if the following conditions are satisfied:

1. If $C \sqsubseteq D \in \mathcal{T}$ then $\text{nnf}(\neg C \sqcup D) \in l_i$ for all $0 \leq i \leq d+1$;
2. $\{A, \neg A\} \not\subseteq l_i$ for every concept name A with $0 \leq i \leq d+1$;
3. If $C_1 \sqcap C_2 \in l_i$ then $\{C_1, C_2\} \subseteq l_i$ for all $0 \leq i \leq d+1$;
4. If $C_1 \sqcup C_2 \in l_i$ then $\{C_1, C_2\} \cap l_i \neq \emptyset$ for all $0 \leq i \leq d+1$;
5. If $\exists R.C \in \lambda_\sigma$ then there is some $1 \leq i \leq d+1$ such that $C \in l_i$ and $R \in r_i$;
6. If $(\leq nS.C) \in \lambda_\sigma$ and there is some $1 \leq i \leq d+1$ such that $S \in r_i$ then $C \in l_i$ or $\dot{C} \in l_i$;
7. If $(\leq nS.C) \in \lambda_\sigma$ and there is some $1 \leq i \leq d+1$ such that $C \in l_i$ and $S \in r_i$ then there is some $1 \leq m \leq n$ such that $\{(\leq mS.C), (\geq mS.C)\} \subseteq \lambda$;
8. For each $1 \leq i \leq d+1$, if $R \in r_i$ and $R \sqsubseteq S$ then $S \in r_i$;
9. If $\forall R.C \in \lambda_\sigma$ and $R \in r_i$ for some $1 \leq i \leq d+1$ then $C \in l_i$;
10. If $\forall R.D \in \lambda_\sigma$, $S \sqsubseteq R$, $\text{Trans}(S)$ and $R \in r_i$ for some $1 \leq i \leq d+1$ then $\forall S.D \in l_i$;
11. If $(\geq nS.C) \in \lambda_\sigma$ then $C_{(\geq nS.C)}^0 \in \lambda_\sigma$ and there are $1 \leq i_1 < \dots < i_n \leq d+1$ such that $\{C, C_{(\geq nS.C)}^j\} \subseteq l_{i_j}$, $S \in r_{i_j}$ for all $1 \leq j \leq n$.
12. If $(\leq nS.C) \in \lambda_\sigma$ and there are no $1 \leq i_1 < \dots < i_{n+1} \leq d+1$ such that $C \in l_{i_j}$ and $S \in r_{i_j}$ for all $1 \leq j \leq n+1$. \triangleleft

Notice that a valid star-type according to Definition 7 is chromatic. If we think of a star-type σ as a node x satisfying σ in a \mathcal{SHOIQ} -forest then σ is valid if no expansion rule is applicable to x . Moreover, due to the conditions 7, 11 and 12 in Definition 7, if there is a term $(\leq nS.D) \in \lambda_\sigma$ for a valid star-type σ then σ has exactly n primary rays $\langle r_i, l_i \rangle, \dots, \langle r_n, l_n \rangle$ w.r.t. $(\leq nS.D)$.

Definition 8 (frame). Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. A frame for $(\mathcal{T}, \mathcal{R})$ is a tuple $\mathcal{F} = (\mathcal{N}_0, \dots, \mathcal{N}_H), \delta, \Phi, \hat{\delta}$, where $H \in \mathbb{N}$ is the dimension of \mathcal{F} ; $\mathcal{N}_i \subseteq \Sigma$ for

all $0 \leq i \leq H$, and all star-types in \mathcal{N}_0 are nominal. We denote $\mathfrak{N} = \bigcup_{i \in \{1, \dots, H\}} \mathcal{N}_i$; δ is a function $\delta : \mathfrak{N} \rightarrow \mathbb{N}$; Φ is a function $\Phi : \mathfrak{N} \rightarrow \Sigma$ where \mathfrak{N} is denoted for the set of all star-types $\sigma \in \mathfrak{N}$ such that one of the three condition holds : (i) σ is nominal; (ii) σ has a ray $\langle r, l \rangle$ such that there is a term $(\leq mR.C)$ with $(\leq mR.C) \in \lambda_\sigma$, $C \in l$ and $R \in r$; (iii) σ has a ray $\langle r, l \rangle$ such that there is a term $(\leq mR.C)$ with $(\leq mR.C) \in l$, $C \in \lambda_\sigma$ and $R \in r$; $\hat{\delta}$ is a function $\hat{\delta} : \Phi(\mathfrak{N}) \rightarrow \mathbb{N}$ which is defined as follows:

For two star-types $\sigma, \sigma' \in \mathfrak{N}$, $\Phi(\sigma) = \Phi(\sigma')$ iff either σ is isomorphic to σ' , or there is a star-type $\omega \in \mathfrak{N} \setminus \mathcal{N}_H$ such that σ and σ' are homomorphic to ω .

Additionally, a star-type $\sigma \in \mathcal{N}_k$ ($0 < k < H$) is linkable with a star-type $\sigma' \in \mathcal{N}_{k-1}$ by a ray $\langle r, l \rangle$ of σ if σ' has a ray $\langle r', l' \rangle$ such that $\lambda_{\sigma'} = l$, $r' = r^-$ and $l' = \lambda_\sigma$ where $r^- = \{R^\ominus \mid R \in r\}$.

Remark 2. For $\sigma, \sigma' \notin \mathcal{N}_H$ and σ, σ' are valid, if σ is homomorphic to σ' then σ is isomorphic to σ' . In fact, if there is a term $(\leq mR.C) \in \lambda_\sigma$ then both σ, σ' have exactly m primary rays w.r.t. $(\leq mR.C)$. If there is a homomorphism between the two sets of primary rays w.r.t. $(\leq mR.C)$ then it is an isomorphism as well.

The frame structure, as introduced in Definition 8, allows us to tile a forest structure by star-types. Such a structure is crucial to obtain termination when designing a tableau-based algorithm. An important difference between a frame and a \mathcal{SHOIQ} -forest is that a frame does not represent nodes corresponding to individuals but stores the number of individuals satisfying a star-type. The function $\delta(\sigma)$ is used for this purpose. In the context of a \mathcal{SHOIQ} -forest, we can think of $\Phi(\sigma)$ as a star-type which is satisfied by nodes forming a set of partitions. In fact, the function Φ maps star-types forming a \mathcal{SHOIQ} -forest into another set of star-types that regroups non-neighbour partitions.

Notice that the function Φ introduced in Definition 8 does not transfer the relation of linkability from \mathfrak{N} to $\Phi(\mathfrak{N})$, and that chromaticity of star-types prevents chromatically linkable star-types from collapsing into a unique star-type by the function Φ . The function $\hat{\delta}(\Phi(\sigma))$ counts the number of partitions that are mapped to a star-type by Φ . Such a function can be defined if all star-types ‘‘covering’’ a partition must be mapped to a unique star-type by Φ . This is a consequence of the Φ 's definition.

Definition 9 (valid frame). Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. A frame $\mathcal{F} = \langle (\mathcal{N}_0, \dots, \mathcal{N}_H), \delta, \Phi, \hat{\delta} \rangle$ is valid if the following conditions are satisfied:

1. For each $\sigma \in \mathfrak{N}$, if $\delta(\sigma) \geq 1$ then σ is valid;
2. For each $\sigma \in \mathfrak{N}$, if $\delta(\sigma) \geq 1$ then $\hat{\delta}(\Phi(\sigma)) \geq 1$;
3. For each $o \in \mathbf{C}_o$ there is a unique $\sigma_o \in \mathcal{N}_0$ such that $o \in \lambda_{\sigma_o}$ and $\delta(\sigma_o) = 1$;
4. For each $\sigma \in \mathfrak{N}$, if $o \in \lambda_\sigma$ with some $o \in \mathbf{C}_o$ then $\Phi(\sigma) = \Phi(\sigma_o)$ and $\hat{\delta}(\Phi(\sigma_o)) = 1$ with $\sigma_o \in \mathcal{N}_0$ such that $o \in \lambda_{\sigma_o}$ and $\delta(\sigma_o) = 1$;
5. For each $0 \leq k < H$ and $\langle \lambda, r, \lambda' \rangle \in 2^{\text{cl}(\mathcal{T}, \mathcal{R})} \times 2^{\mathbf{R}(\mathcal{T}, \mathcal{R})} \times 2^{\text{cl}(\mathcal{T}, \mathcal{R})}$ with $r^- = \{R^\ominus \mid R \in r\}$,

$$\sum_{\sigma \in \mathcal{N}_k} \delta(\sigma) |\bar{\mu}_\sigma|_{\langle \lambda, r, \lambda' \rangle} = \sum_{\sigma' \in \mathcal{N}_{k+1}} \delta(\sigma') |\bar{\nu}_{\sigma'}|_{\langle \lambda', r^-, \lambda \rangle}$$

- where $|\bar{\nu}_\sigma|_{\langle \lambda, r, \lambda' \rangle}$ and $|\bar{\mu}_\sigma|_{\langle \lambda, r, \lambda' \rangle}$ are denoted for the number of components $\langle r', l' \rangle$ of respective $\bar{\nu}_\sigma$ and $\bar{\mu}_\sigma$ such that $\lambda_\sigma = \lambda$, $r' = r$ and $l' = \lambda'$;
6. For each $\langle \lambda, r, \lambda' \rangle \in 2^{\text{cl}(\mathcal{T}, \mathcal{R})} \times 2^{\mathbf{R}(\mathcal{T}, \mathcal{R})} \times 2^{\text{cl}(\mathcal{T}, \mathcal{R})}$ with $r^- = \{R^\ominus \mid R \in r\}$,

$$\sum_{\sigma \in \bar{\mathfrak{N}}} \widehat{\delta}(\Phi(\sigma)) | \bar{\xi}_{\Phi(\sigma)} |_{\langle \lambda, r, \lambda' \rangle} = \sum_{\sigma' \in \bar{\mathfrak{N}}} \widehat{\delta}(\Phi(\sigma')) | \bar{\xi}_{\Phi(\sigma')} |_{\langle \lambda', r^-, \lambda \rangle}$$

where $|\bar{\xi}_{\Phi(\sigma)}|_{\langle \lambda, r, \lambda' \rangle} = |\bar{\nu}_{\Phi(\sigma)}|_{\langle \lambda, r, \lambda' \rangle} + |\bar{\mu}_{\Phi(\sigma)}|_{\langle \lambda, r, \lambda' \rangle}$ ◁

Remark 3. It is not required that star-types $\Phi(\sigma)$ are valid. We will use function Φ to trim rays $\langle r, l \rangle$ of star-types such that (i) $\langle r, l \rangle$ or $\langle r^-, l \rangle$ is not a primary ray of every star-type. The images of star-types σ by Φ , i.e. trimmed star-types $\Phi(\sigma)$, are employed to represent partitions obtained from merge processes. As described in Section 3, in order to govern partitions, it suffices to deal with nodes x containing a term ($\leq mR.C$) and the R -neighbours of x containing C . For this reason, the function Φ maps only star-types in $\bar{\mathfrak{N}}$ by trimming non-primary rays.

The notion of validity for a frame is crucial to establish a connection with the tableau-based algorithm presented in Section 3, i.e., how to build a \mathcal{SHOIQ} -forest from a valid frame, and inversely. The condition 1 in Definition 9 requires that every star-type satisfied by at least one node must be valid. The condition 2 implies that each valid star-type including a primary ray will be mapped by Φ . The condition 3 ensures that each nominal is counted exactly once while the condition 4 imposes that all nominal star-types containing some $o \in \mathbf{C}_o$ are mapped into a unique star-type by Φ . In the context of a \mathcal{SHOIQ} -forest, these conditions imply that for each nominal $o \in \mathbf{C}_o$ there is exactly one tree whose root contains o and there is exactly one partition containing o . The condition 5 allows for linking star-types at level k with star-types at level $k - 1$ and $k + 1$. It ensures that each node x satisfying (or counted for) a star-type σ at level k is linked by its rays to neighbours satisfying star-types at level $k - 1$ and $k + 1$. The number of these neighbours corresponds exactly to the number of σ 's rays. Finally, the condition 6 deals with partitions. In the context of a \mathcal{SHOIQ} -forest where $\Phi(\sigma)$ represents the image of a set of partitions, the condition 6 points out how star-types $\Phi(\sigma)$ would be interconnected.

Lemma 2. *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base.*

1. *If the tableau algorithm can build a clash-free and complete \mathcal{SHOIQ} -forest for $(\mathcal{T}, \mathcal{R})$ then there is a valid frame for $(\mathcal{T}, \mathcal{R})$.*
2. *If there is a valid frame $\mathcal{F} = \langle (\mathcal{N}_0, \dots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta} \rangle$ for $(\mathcal{T}, \mathcal{R})$ then the tableau algorithm can build a clash-free and complete \mathcal{SHOIQ} -forest for $(\mathcal{T}, \mathcal{R})$.*

Lemma 2 points out the equivalence between a clash-free and complete \mathcal{SHOIQ} -forest and a valid frame for $(\mathcal{T}, \mathcal{R})$. The following lemma affirms that there is an exponential structure, a valid frame, which can represent a \mathcal{SHOIQ} -forest whose size may be doubly exponential.

Lemma 3. *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. The size of a valid frame $\mathcal{F} = \langle (\mathcal{N}_0, \dots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta} \rangle$ is bounded by an exponential function in the size of $(\mathcal{T}, \mathcal{R})$.*

We can sketch a proof of the lemma here. We have $H \leq K$ where $K = 2^{2m+k} \times 2$ with $m = \text{card}\{\text{cl}(\mathcal{T}, \mathcal{R})\}$ and $k = \text{card}\{\mathbf{R}_{(\mathcal{T}, \mathcal{R})}\}$. Moreover, each star-type has at most M distinct rays where $M = \sum m_i + E$, m_i occurs in a number restriction term ($\geq m_i R.C$) appearing in \mathcal{T} , and E is the number of distinct terms $\exists R.C$ appearing in \mathcal{T} . If we denote Σ for the set of all star-types then $\text{card}\{\Sigma\} \leq ((\text{card}\{\text{cl}(\mathcal{T}, \mathcal{R})\})^2 \times \text{card}\{\mathbf{R}_{(\mathcal{T}, \mathcal{R})}\})^M$. Since $\delta(\sigma)$ is bounded by $M\delta(\sigma')$ where σ', σ are respective star-types at level $k-1$ and k , it holds that $\delta(\sigma) \leq M^{2^{2m+k} \times 2}$. If $\delta(\sigma)$ is represented as a binary number then it takes an exponential number of bits.

Based on Lemma 3 and 2, we can present straightforwardly an optimal worst-case algorithm for checking the consistency of a *SHOIQ* knowledge base. However, such an algorithm cannot be used in practice since the non-determinism is not sufficiently constrained to obtain termination in feasible time. In the sequel, based on the results obtained so far, we try to design an algorithm which has more goal-directed behaviour.

Blocking Condition for a Frame Let $\mathcal{F} = \langle (\mathcal{N}_0, \dots, \mathcal{N}_H), \delta, \Phi, \hat{\delta} \rangle$ be a frame. A star-type $\sigma_k \in \mathcal{N}_k$ with $0 < k \leq H$ is *blocked* if there are $\sigma_i \in \mathcal{N}_i$ with $0 \leq i \leq k$ such that σ_i is linkable with σ_{i-1} for all $i \in \{1, \dots, k\}$, then there are $0 < k_1 < k_2 < k_3 < k_4 \leq k$ such that:

1. $\lambda_{\sigma_{k_1}} = \lambda_{\sigma_{k_2}}, \bar{\nu}_{\sigma_{k_1}} = \bar{\nu}_{\sigma_{k_2}}$, and there is no $0 < j < k_2$ such that $j \neq k_1, \lambda_{\sigma_j} = \lambda_{\sigma_{k_2}}$ and $\bar{\nu}_{\sigma_j} = \bar{\nu}_{\sigma_{k_2}}$;
2. $\lambda_{\sigma_{k_3}} = \lambda_{\sigma_{k_4}}, \bar{\nu}_{\sigma_{k_3}} = \bar{\nu}_{\sigma_{k_4}}$, and there is no $k_2 < j < k_4$ such that $j \neq k_3, \lambda_{\sigma_j} = \lambda_{\sigma_{k_4}}$ and $\bar{\nu}_{\sigma_j} = \bar{\nu}_{\sigma_{k_4}}$.

Notice that this blocking condition is looser than the blocking condition introduced in Definition 5 for a *SHOIQ*-forest. Since we cannot determine the path from root to a node satisfying a star-type over a frame, it is not possible to check blocking condition in the same way as for a *SHOIQ*-forest. The blocking condition for a frame, as described above, implies that a node satisfying a blocked star-type must have an ancestor which is blocked according to the blocking condition for a *SHOIQ*-forest.

We are now ready to propose an EXPSPACE tableau-based algorithm for *SHOIQ*. It starts by generating nominal star-types at level 0. The goal of the algorithm is to replace progressively non-valid star-types with those which are “nearer” the validity. When a non-valid star-type σ at a level h is replaced with a “better” one σ' at the same level, the algorithm adds $\delta(\sigma)$ to $\delta(\sigma')$ and sets $\delta(\sigma) = 0$. This may lead to update $\delta(\omega)$ where ω is linkable with σ . In addition, the algorithm has to maintain two functions $\Phi(\sigma)$ and $\hat{\delta}(\Phi(\sigma))$ such that the conditions 5 and 6 in Definition 9 always hold after each update of star-types.

An important difference between the tableau algorithm presented in Section 3 and the tableau algorithm for constructing a valid frame is that the latter adds star-types to a frame and updates functions $\delta(\sigma)$ and $\hat{\delta}(\Phi(\sigma))$ instead of adding nodes for representing individuals. Again, we refer the readers to [5] where the rules for building a valid frame, called *frame rules*, can be found.

Soundness of the tableau-based algorithm for building a frame can be established thanks to Lemma 2. Since each frame rule has its counterpart in the expansion rules, completeness of the algorithm can be shown by using the same arguments as those employed to prove Lemma 1. From these results and Lemma 3, we obtain the following main result of the section:

Theorem 2. *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. The tableau algorithm for constructing a frame is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$ and it runs in EXPSpace in the size of $(\mathcal{T}, \mathcal{R})$.*

5 Conclusion and Discussion

We have presented in this paper a practical EXPSpace decision procedure for the logic \mathcal{SHOIQ} . The construction of this algorithm is founded on the well-known results for \mathcal{SHOIQ} and \mathcal{C}^2 . First, we have based our approach on a technique that constructs tree-like structures for representing a model. This allows us to reuse the standard blocking technique over these tree-like structures to obtain termination. Second, we have transferred to \mathcal{SHOIQ} the method used for constructing a NEXPTIME algorithm for \mathcal{C}^2 . This enables us to represent a doubly exponential \mathcal{SHOIQ} -forest by an exponential structure.

The tableau algorithms proposed in the present paper have introduced several non-deterministic rules, e.g., \bowtie - or \leq_o -rules. In particular, the most non-deterministic behaviour of the tableau algorithm for building a valid frame is to update the function δ for star-types ω when applying frame rules to a star-type σ which is linkable with ω . Such an update may lead to choose a subset from an exponential set of linkable star-types. An open issue consists in investigating whether the complexity resulting from this behaviour is comparable to that caused by the \leq - and \leq_o -rules.

Acknowledgements. Thanks to the reviewers for their helpful comments.

References

1. Horrocks, I.: The FaCT system. In: de Swart, H. (ed.) Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98). Lecture Notes in Artificial Intelligence, vol. 1397, pp. 307–312. Springer (1998), [download/1998/Horr98b.pdf](http://www.springer.com/978-3-540-65198-8)
2. Horrocks, I., Sattler, U.: A tableau decision procedure for \mathcal{SHOIQ} . *Journal Of Automated Reasoning* 39(3), 249–276 (2007)
3. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 1999). Springer (1999)
4. Kazakov, Y., Motik, B.: A resolution-based decision procedure for \mathcal{SHOIQ} . *Journal of Automated Reasoning* 40(2–3), 89–116 (2008)
5. Le Duc, C., Lamolle, M., Curé, O.: An EXPSpace tableau-based algorithm for \mathcal{SHOIQ} . In: Technical Report. <http://www.iut.univ-paris8.fr/~leduc/papers/RR2012a.pdf> (2012)
6. Patel-Schneider, P., Hayes, P., Horrocks, I.: OWL web ontology language semantics and abstract syntax. In: W3C Recommendation (2004)
7. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information* 14(3), 369–395 (2005)
8. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical OWL-DL reasoner. *Journal of Web Semantics* 5(2), 51–53 (2007)
9. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research* 12, 199–217 (2000)

Role-depth Bounded Least Common Subsumers for \mathcal{EL}^+ and \mathcal{ELI}

Andreas Ecke and Anni-Yasmin Turhan*

TU Dresden, Institute for Theoretical Computer Science

Abstract. For \mathcal{EL} the least common subsumer (lcs) need not exist, if computed w.r.t. general TBoxes. In case the role-depth of the lcs concept description is bounded, an approximate solution can be obtained. In this paper we extend the completion-based method for computing such approximate solutions to \mathcal{ELI} and \mathcal{EL}^+ . For \mathcal{ELI} the extension needs to be able to treat complex node labels. For \mathcal{EL}^+ a naive method generates highly redundant concept descriptions for which we devise a heuristic that produces smaller, but equivalent concept descriptions. We demonstrate the usefulness of this heuristic by an evaluation.

1 Introduction

The reasoning service least common subsumer (lcs) computes a concept description from set of concept descriptions expressed in a DL \mathcal{L} . The resulting concept description subsumes all of the input concept descriptions and is the least w.r.t. subsumption expressible in \mathcal{L} to do so. This reasoning service has turned out to be useful for the augmentation of TBoxes [1] and as a subtask when computing the (dis)similarity of concept descriptions [2, 3] or other non-standard inferences.

In particular several bio-medical TBoxes are written in extensions of \mathcal{EL} that allow to model roles in a more detailed way, such as SNOMED [4] which allows to use role inclusions and is written in \mathcal{ELH} or the Gene Ontology [5] and the FMA ontology [6] which are both written in \mathcal{EL}^+ , which is a DL that extends \mathcal{ELH} by right identities for roles. For these extensions of \mathcal{EL} standard DL reasoning can still be done in polynomial time [7]. However, the GALEN ontology uses the DL $\mathcal{ELHI}f_{\mathcal{R}^+}$ —a DL with inverse roles, which are known to make subsumption w.r.t. general TBoxes EXPTIME-complete [7] due to the use of inverse roles. These TBoxes are known to be very large and are mostly build by hand.

If computed w.r.t. general or just cyclic \mathcal{EL} -TBoxes, the lcs need not exist [8], since resulting cyclic concept descriptions cannot be expressed in \mathcal{EL} . In [9] an extension of \mathcal{EL} by fixed-points has been investigated that can capture such concept descriptions. Since we want to obtain a concept description for the lcs that is expressed in that DL in which the TBox is written, we follow the idea from [10] and compute as an approximative solution the *role-depth bounded lcs*: k -lcs which has a maximal nesting of quantifiers limited to k .

* Partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

Table 1. Constructors and axioms for \mathcal{EL} and some of its extensions

Name	Syntax	Semantics
top	\top	$\Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
inverse role	r^{-}	$\{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}}\}$
general concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion axiom	$r_1 \circ \dots \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \dots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$

The approach to compute the k -lcs is to employ the completion method that is used to classify the TBox. This method builds a graph structure, which is saturated by completion rules [11, 7]. In case of \mathcal{EL} the k -lcs can be more or less directly be read off from the saturated completion graph. In this paper we devise computation algorithms for the k -LCS for the DLs \mathcal{EL}^+ and in \mathcal{ELI} . It turns out that for \mathcal{EL}^+ the computation algorithm is the same as for \mathcal{EL} [10]. While the polynomial time completion algorithm for \mathcal{EL}^+ works on graph structures with static node sets and have simple labellings, the algorithm for \mathcal{ELI} requires dynamic nodes sets and uses complex labels. In [12] such a completion algorithm for \mathcal{ELI} has been devised, which we employ for the computation of the k -lcs in \mathcal{ELI} .

For both methods we show that the obtained concept is a common subsumer and that it is minimal w.r.t. subsumption for the given role-depth bound k . Thus, the obtained concept description is the *exact* lcs, if the exact lcs exists for a role-depth n and the k -lcs is computed for a maximal role-depth of $k \geq n$.

The concept descriptions obtained in this way turn out to be highly redundant. In order to obtain concise and readable concept descriptions, we devise a heuristic to obtain smaller, equivalent concept descriptions.

This paper is organised as follows: next, we introduce the basic notions. In Section 3 we recall the completion algorithm for \mathcal{EL}^+ and devise the computation algorithms for the k -lcs in \mathcal{EL}^+ . The computation algorithm for \mathcal{ELI} is presented in Section 4. In Section 5 we present the simplification heuristic to obtain smaller \mathcal{EL}^+ -concept descriptions. We end with conclusions and remarks on future work.

2 Preliminaries

We assume that the reader is familiar with the basic notions of DLs, for an introduction see [13]. We introduce the DLs used in this paper formally. *Concept descriptions* are inductively defined from a set of *concepts names* N_C and a set of *role names* N_R by applying the constructors from the upper half of Table 1. In particular, \mathcal{EL} -concept descriptions only allow for conjunctions, existential restrictions, and the top concept \top . \mathcal{EL}^+ additionally allows for complex *role inclusion axioms* (RIAs). These role inclusions can express role hierarchies ($s \sqsubseteq r$) and transitive roles ($r \circ r \sqsubseteq r$). The semantics are displayed in the lower half

of Table 1. \mathcal{ELI} -concept description extend \mathcal{EL} -concept descriptions by the use of *inverse roles*.

The concept constructors and axioms are interpreted in the standard way. We denote by $N_{C,\mathcal{T}}$ and $N_{R,\mathcal{T}}$ the sets of concept names and role names that occur in a TBox \mathcal{T} . For a concept description C we denote by $rd(C)$ its role-depth, i.e., its maximal nesting of quantifiers. We define the central reasoning services of this paper.

Definition 1 ((Role-depth bounded) least common subsumer). *Let \mathcal{L} be a DL, \mathcal{T} be a \mathcal{L} -TBox and C_1, \dots, C_n be \mathcal{L} -concept descriptions. Then the \mathcal{L} -concept description D is the least common subsumer of C_1, \dots, C_n w.r.t. \mathcal{T} iff (1) $C_i \sqsubseteq_{\mathcal{T}} D$ for all $i \in \{1, \dots, n\}$, and (2) for all \mathcal{L} -concept descriptions E : $C_i \sqsubseteq_{\mathcal{T}} E$ for all $i \in \{1, \dots, n\}$ implies $D \sqsubseteq_{\mathcal{T}} E$.*

Let $k \in \mathbb{N}$. Then the \mathcal{L} -concept description D is the role-depth bounded least common subsumer of C_1, \dots, C_n w.r.t. \mathcal{T} and the role-depth k (k -lcs(C_1, \dots, C_n)) iff (1) $rd(D) \leq k$, (2) $C_i \sqsubseteq_{\mathcal{T}} D$ for all $i \in \{1, \dots, n\}$, and (3) for all \mathcal{L} -concept descriptions E with $rd(E) \leq k$: $C_i \sqsubseteq_{\mathcal{T}} E \forall i \in \{1, \dots, n\}$ implies $D \sqsubseteq_{\mathcal{T}} E$.

For the DLs considered in this paper the (k -)lcs is unique up to equivalence, thus we speak of the (k -)lcs.

3 Computing the k -lcs in \mathcal{EL}^+

The algorithms to compute the role-depth bounded lcs rely on completion graphs produced by completion-based subsumption algorithms. Completion algorithms work on normalized TBoxes and for which they build a completion graph and exhaustively apply completion rules. After this step, the completion graph contains all subsumption relations from the TBox explicitly.

3.1 Completion Algorithm for \mathcal{EL}^+

An \mathcal{EL}^+ -TBox \mathcal{T} is in normal form, if all concept inclusions in \mathcal{T} are of the form $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$, or $\exists r.A \sqsubseteq B$ with $A, A_1, A_2, B \in N_C$ and $r \in N_R$; and all role inclusions are of the form $s \sqsubseteq r$ or $s \circ t \sqsubseteq r$ with $\{r, s, t\} \subseteq N_R$. All \mathcal{EL}^+ -TBoxes can be normalized by applying a set of normalization rules [11].

The completion graph for a normalized TBox \mathcal{T}' used by the completion algorithm is of the form (V, E, S) , where $V = N_{C,\mathcal{T}'} \cup \{\top\}$ is the set of nodes, $E \subseteq V \times N_{R,\mathcal{T}'} \times V$ is the set of role name labeled edges and $S : V \rightarrow 2^{N_{C,\mathcal{T}'} \cup \{\top\}}$ is the node-labeling. The completion algorithms starts with an initial graph (V, E, S) with $E = \emptyset$ and $S(A) = \{A, \top\}$ for each $A \in N_{C,\mathcal{T}'} \cup \{\top\}$ and exhaustively applies a set of completion rules from [11] until no more rule applies.

Once the rule-applications finished, all subsumption relations can be directly be read off the completion graph. This completion algorithm is sound and complete as shown in [11]. Specifically, given a normalized \mathcal{EL}^+ -TBox \mathcal{T} and its completion graph (V, E, S) after all completions rules were applied exhaustively, we have for each $A, B \in V$ and $r \in E$:

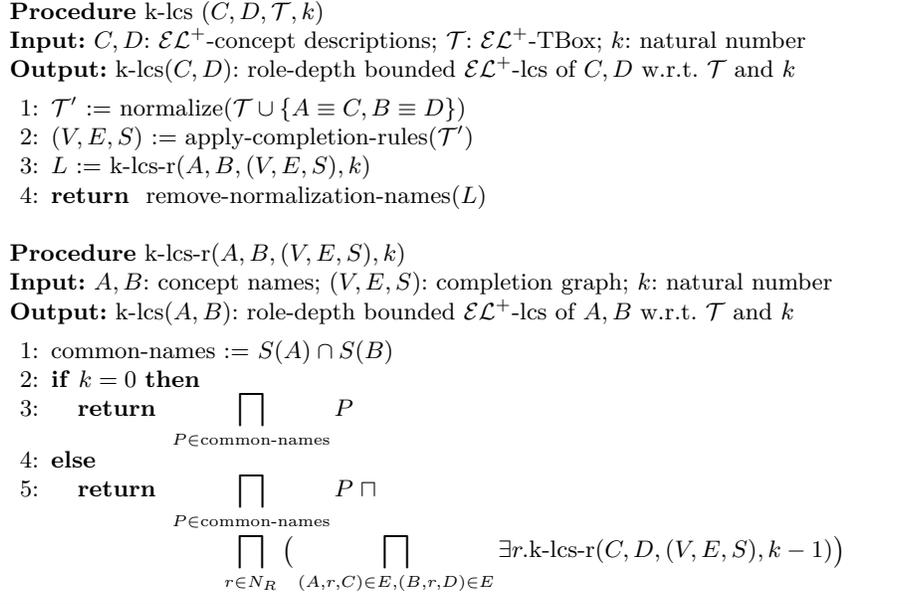


Fig. 1. Computation Algorithm for role-depth bounded \mathcal{EL}^+ -lcs.

Soundness If $B \in S(A)$, then $A \sqsubseteq_{\mathcal{T}} B$; and
if $(A, r, B) \in E$, then $A \sqsubseteq_{\mathcal{T}} \exists r.B$.

Completeness If $A \sqsubseteq_{\mathcal{T}} B$, then $B \in S(A)$; and
if $A \sqsubseteq_{\mathcal{T}} \exists r.B$, then there are $C, D \in V$ with $C \in S(A)$, $B \in S(D)$ and
 $(C, r, D) \in E$.

3.2 Computation Algorithm of the k-lcs in \mathcal{EL}^+

The resulting completion graph can be used to compute the role-depth bounded lcs. All RIAs from the \mathcal{EL}^+ -TBox are explicitly captured in the completion graph in the following sense: for each edge in the completion graph labeled with some role r , the completion algorithm also creates edges for all its super-roles. This means that for computing the k-lcs for an \mathcal{EL}^+ -TBox the same algorithm can be used as for \mathcal{EL} , which was introduced in [10] and is shown in Algorithm 3.2 for the binary lcs. The idea is to introduce new concept names for the concept descriptions of interest and to apply the completion algorithm. Then, starting from the newly introduced names A and B , traverse the completion graph simultaneously. More precisely, for the tree unravelings of depth k for A and B the cross product is computed. In a post-processing step those concept names have to be removed from the concept that were introduced during normalization. Obviously, this method creates heavily redundant concept descriptions, due to the multiple edge labellings due to RIAs.

```

Procedure simplify( $C, (V, E, S), \mathcal{T}$ )
Input:  $C$ :  $\mathcal{EL}^+$ -concept description;  $(V, E, S)$ : completion graph;  $\mathcal{T}$ :  $\mathcal{EL}^+$ -TBox
Output: simplify( $C$ ): simplified concept description
1: Let  $C \equiv A_1 \sqcap \dots \sqcap A_n \sqcap \exists r_1.D_1 \sqcap \dots \sqcap \exists r_m.D_m$  with  $A_i \in N_C$  for  $1 \leq i \leq n$ .
2:  $Conj := \{A_i \mid 1 \leq i \leq n\} \cup \{\exists r_j.D_j \mid 1 \leq j \leq m\}$ 
3: for all  $X \in Conj$  do
4:   for all  $Y \in Conj$  do
5:     if  $X \neq Y \wedge \text{subsumes-H}(X, Y, (V, E, S), \mathcal{T})$  then
6:        $Conj := Conj \setminus \{X\}$ 
7:     break
8: for all  $X \in Conj$  do
9:   if  $X = \exists r_j.D_j$  then
10:     $Conj := (Conj \setminus \{\exists r_j.D_j\}) \cup \{\exists r_j.\text{simplify}(D_j, (V, E, S), \mathcal{T})\}$ 
11: return  $\bigcap_{X \in Conj} X$ 

```

Fig. 2. Simplification algorithm for \mathcal{EL}^+ -concept descriptions

3.3 Simplifying \mathcal{EL}^+ -Concept Descriptions

The highly redundant \mathcal{ELH} -concept descriptions obtained from the k -lcs algorithm, need to be simplified, in order to make the resulting concept description readable. The general idea for the simplification is to remove those subtrees from the syntax tree which are subsumers of any of their sibling subtrees. For a conjunction of concept names, this results in the least ones (w.r.t. $\sqsubseteq_{\mathcal{T}}$).

Algorithm 2 computes the simplification of an \mathcal{EL}^+ -concept description. Note, that the algorithm needs to be applied after the normalization names were removed, otherwise it might remove names from the original TBox that subsume normalization names, which get removed later during denormalization.

For the soundness of the simplification procedure `simplify`, it is only necessary to ensure that the procedure ‘`subsumes-H`’ is sound. However, for our purpose this procedure does not have to be complete. This might result in simplifications that are correct k -lcs, but that are still redundant. This heuristic is given in [14]. The idea is to make simple structural comparison depending on the concept constructor of the concepts in question.

Obviously, it would be desirable to avoid the generation of highly redundant concept descriptions, instead of reducing them in a post-processing step. Due to interactions with denormalization, such optimizations need to be conservative. Such optimizations have been investigated in [14], which avoid unnecessary branching and role-depth of the generated concept description. Interestingly, these optimizations do not only speed-up the execution of Algorithm 3.2, but also of the subsequent simplification, see [14].

Evaluation. The k -lcs algorithm and the simplification algorithm are implemented in our system GEL¹, which is implemented on top of the jCEL rea-

¹ GEL is freely available from <http://sourceforge.net/p/gen-el>.

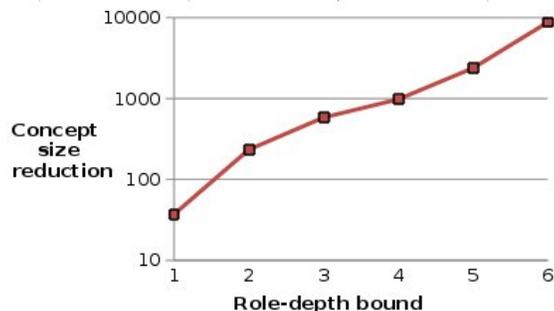


Fig. 3. Average gain in concept size for simplified k -lcs computed w.r.t. NOT-GALEN

soner² [15]. We have tested the effectiveness of the simplification procedure on the NotGalen ontology, which is a version of the GALEN ontology pruned to \mathcal{EL}^+ . Some input concept pairs resulted in run-times over a minute for $k = 6$, which were mostly dominated by the run-time of the k -lcs-r-procedure. Simplification of larger concepts was faster by a factor of 10 or more. Figure 3 shows the average gain in concept size by simplification on various input pairs for different values of k . For $k = 6$ concepts with a size of several thousands were reduced to a concept size of 30 to 40, which are large, but still readable concept descriptions. In an extreme case a concept of size of over 10^6 was reduced to a size of 140. For more empirical results and details on the implementation of GEL see [14].

4 Computing the k -lcs in \mathcal{ELI}

To handle inverse roles correctly, the completion algorithm needs to be adapted in several ways. The normal form for TBoxes is the same as before.

4.1 Completion Algorithm for \mathcal{ELI}

The \mathcal{EL} -completion algorithm has been extended to \mathcal{ELI} in [12]. One adaptation is that the node set V is not fixed. Consider the example TBox $\mathcal{T} = \{\exists r^-.A \sqsubseteq C, A \sqsubseteq \exists r.B\}$. In this TBox, A has an r -successor subsumed by B and each r -predecessor A implies C . However, that does not mean that C is also a subsumer of B – only those elements in $B^{\mathcal{I}}$, that are r -successors of elements in $A^{\mathcal{I}}$ are also in $C^{\mathcal{I}}$. Thus, $C \not\subseteq S(B)$. On the other hand we know that $A \sqsubseteq \exists r.C$. To solve this problem, we need to have a dynamic node set V , add a new node u to V for $u = B \sqcap \exists r^-.A$ and then add C to the completion set $S(u)$.

The node set V is defined as $V \subseteq N_{C,\mathcal{T}} \times 2^{\{\exists r.X \mid r \text{ is a role, } X \in N_{C,\mathcal{T}}\}}$. A node A with $A \in N_{C,\mathcal{T}}$ from the node set for \mathcal{EL}^+ would then correspond to the node (A, \emptyset) from the node set for \mathcal{ELI} . We will formalize the meaning of nodes in the node set V by defining the concept descriptions that these nodes correspond to:

² jCEL is freely available from <http://jcel.sourceforge.net>.

Definition 2 (Concept descriptions for nodes). Let \mathcal{T} be a normalized \mathcal{ELI} -TBox and (V, E, S) its completion graph. Then we define for each node $u = (A, \phi) \in V$: $vconcept(u) = A \sqcap \prod_{\exists r.X \in \phi} \exists r.X$

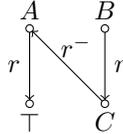
The graph (V, E, S) for the completion algorithm for \mathcal{ELI} starts with $V = \{(A, \emptyset) \mid A \in N_{C, \mathcal{T}}\}$, $E = \emptyset$ and $S((A, \emptyset)) = \{A, \top\}$ for all $A \in N_{C, \mathcal{T}}$. The completions rules for \mathcal{ELI} are the following:

- CI1** If $A_1 \in S(v)$ and $A_1 \sqsubseteq B \in \mathcal{T}$ and $B \notin S(v)$,
then $S(v) := S(v) \cup \{B\}$
- CI2** If $A_1, A_2 \in S(v)$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $B \notin S(v)$,
then $S(v) := S(v) \cup \{B\}$
- CI3** If $A_1 \in S(u)$, $v = (B, \emptyset)$ and $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$ and $(u, r, v) \notin E$,
then $E := E \cup \{(u, r, v)\}$
- CI4** If $(u, r, v) \in E$, $B_1 \in S(v)$ and $\exists r.B_1 \sqsubseteq C \in \mathcal{T}$ and $C \notin S(u)$,
then $S(u) := S(u) \cup \{C\}$
- CI5** If $(u, r, v) \in E$, $v = (B, \psi)$, $A_1 \in S(u)$, $\exists r^-.A_1 \sqsubseteq B_1 \in \mathcal{T}$ and $B_1 \notin S(v)$,
then
 $v' := (B, \psi \cup \{\exists r^-.A_1\})$
if $v' \notin V$ then $V := V \cup \{v'\}$, $E := E \cup \{(u, r, v')\}$, $S(v') := S(v) \cup \{B_1\}$
else $E := E \cup \{(u, r, v')\}$, $S(v') := S(v') \cup \{B_1\}$

The completion algorithm for \mathcal{ELI} defined this way is again sound. For completeness one needs to consider only those edges that do not point to nodes, which have an ‘extended copy’ generated by rule CI5, i.e., edges (u, r, v) for which there is no $\exists r^-.A \sqsubseteq B \in \mathcal{T}$ with $A \in S(u)$ and $B \notin S(v)$. We call those edges *bad edges* and collect them in the bad edge set E_{bad} . However, since for each edge (u, r, v) in E_{bad} there is $(u, r, v') \in E \setminus E_{bad}$ with $vconcept(v') \sqsubseteq_{\mathcal{T}} vconcept(v)$, completeness for *good* edges is sufficient to show that the concept description obtained by Algorithm 1 is a common subsumer [16].

4.2 Computation of the k-lcs in \mathcal{ELI}

Since \mathcal{ELI} allows for inverse roles, we may also traverse edges backwards (i.e., use the inverse role of the role that the edge is labeled with in the k -lcs concept description). However, we can only traverse those edges backwards, that we just came from—as you can see in the example for $\mathcal{T} = \{A \sqsubseteq \exists r.\top, B \sqsubseteq \exists r.C, C \sqsubseteq \exists r^-.A\}$, which results in the following completion graph:



Now, traversing this completion graph to compute the lcs of A and B without going backwards, we would get the result $\top \sqcap \exists r.\top$ and then get stuck in the \top node. However, the lcs of A and B is $\exists r.\exists r^-.A$, therefore the algorithm must to go backwards from \top to A using the edge (A, r, \top) as (\top, r^-, A) , which yields

the correct lcs. To see that the algorithm may not go backwards along arbitrary edges consider to go from A to C using the edge (C, r^-, A) as (A, r, C) . This would clearly be wrong, since we don't have $A \sqsubseteq_{\mathcal{T}} \exists r.C$. Thus the algorithm may only traverse backwards on those edges that led to the current node.

Therefore, the recursive algorithm needs to know not only the current nodes, but also the whole path from the start to the current node. This path is given in the form $[u_0, r_1, u_1, r_2, \dots, r_n, u_n]$ where u_0 is the starting node, u_n the current node, and $(u_{i-1}, r_i, u_i) \in E$ are edges of the completion graph that have been traversed. For each path $[u_0, r_1, u_1, r_2, \dots, r_n, u_n]$ we will define the concept description they correspond to.

Definition 3 (Concept descriptions for paths). *Let \mathcal{T} be a normalized \mathcal{ELI} -TBox and (V, E, S) its completion graph. Then we define for each path $l = [u_0, r_1, u_1, r_2, \dots, r_n, u_n]$*

$$lconcept(l) = vconcept(u_n) \sqcap \exists r_n^-. (vconcept(u_{n-1}) \sqcap \exists r_{n-1}^- . (\dots \sqcap \exists r_1^- . vconcept(u_0) \dots))$$

Algorithm 1 depicted below computes the role-depth bounded lcs for two \mathcal{ELI} -concept descriptions C and D w.r.t. a general \mathcal{ELI} -TBox. This algorithm differs from the Algorithm 3.2 for \mathcal{EL}^+ mainly only in the following aspects:

- Algorithm 1 uses the whole path to the current node instead of the node itself.
- While in Algorithm 3.2 the nodes to visit from the current node are computed implicitly, Algorithm 1 stores all successors of the paths p_1 and p_2 explicitly in the sets S_1 and S_2 .
- Both algorithms traverse all edges (u, r, v) from the current node u , but Algorithm 1 additionally traverses the last edge backwards, if it is the inverse of r .

We give a proof sketch that Algorithm 1 indeed computes the k -lcs. Condition (1) from the Definition of the role-depth bounded lcs is obviously given.

Common Subsumer. The fact that Algorithm 1 yields a common subsumer follows directly from the following lemma:

Lemma 1. *Let $L = k\text{-lcs-r}(p_1, p_2, (V, E, S), k)$ for the two given paths $p_1 = [u_0, r_1, u_1, r_2, \dots, r_n, u_n]$ and $p_2 = [v_0, s_1, v_1, s_2, \dots, s_m, v_m]$. Then $lconcept(p_1) \sqsubseteq_{\mathcal{T}} L$ and $lconcept(p_2) \sqsubseteq_{\mathcal{T}} L$.*

Proof. This lemma can be proven by induction on the role-depth k of L . For $k = 0$, $L = A_1 \sqcap A_2 \sqcap \dots \sqcap A_l$ must be a conjunction of concept names $A_i \in S(u_n) \cap S(v_m)$, $0 \leq i \leq l$. Then soundness of the completion algorithm yields that for each A_i , we have $lconcept(p_1) \sqsubseteq_{\mathcal{T}} vconcept(u_n) \sqsubseteq_{\mathcal{T}} A_i$ and similarly $lconcept(p_2) \sqsubseteq_{\mathcal{T}} A_i$; therefore $lconcept(p_1) \sqsubseteq_{\mathcal{T}} L$ and $lconcept(p_2) \sqsubseteq_{\mathcal{T}} L$.

For $k \geq 1$, L is a conjunction of concept names and existential restrictions. For concept names, the same argument as above holds. All existential restrictions are of the form $\exists r.k\text{-lcs-r}(l_1, l_2, (V, E, S), k - 1)$ where l_1 is either

Algorithm 1 Computation of a role-depth bounded \mathcal{ELI} -lcs.

Procedure $k\text{-lcs}(C, D, \mathcal{T}, k)$

Input: C, D : \mathcal{ELI} -concept descriptions; \mathcal{T} : \mathcal{ELI} -TBox; k : natural number

Output: $k\text{-lcs}(C, D)$: role-depth bounded \mathcal{ELI} -lcs of C and D w.r.t. \mathcal{T} and k

- 1: $\mathcal{T}' := \text{normalize}(\mathcal{T} \cup \{A \equiv C, B \equiv D\})$
- 2: $(V, E, S) := \text{apply-completion-rules}(\mathcal{T}')$
- 3: $L := k\text{-lcs-r}([(A, \emptyset)], [(B, \emptyset)], (V, E, S), k)$
- 4: **return** $\text{remove-normalization-names}(L)$

Procedure $k\text{-lcs-r}(p_1, p_2, (V, E, S), k)$

Input: $p_1 = [(A_0, \emptyset), r_1, \dots, r_n, (A_n, \phi_n)]$ and $p_2 = [(B_0, \emptyset), s_1, \dots, s_m, (B_m, \psi_m)]$: two paths in the completion graph; (V, E, S) : completion graph; k : natural number

Output: role-depth bounded \mathcal{ELI} -lcs of $\text{lconcept}(p_1)$ and $\text{lconcept}(p_2)$ w.r.t. \mathcal{T} and k

- 1: $\text{result-concept} := \prod_{C \in S((A_n, \phi_n)) \cap S((B_m, \psi_m))} C$
 - 2: **if** $k > 0$ **then**
 - 3: **for all** $r \in N_R$ **do**
 - 4: $S_1 := \{[(A_0, \emptyset), r_1, \dots, r_n, (A_n, \phi_n), r, (A, \phi)] \mid ((A_n, \phi_n), r, (A, \phi)) \in E\}$
 - 5: **if** $n > 0 \wedge r = r_n^-$ **then**
 - 6: $S_1 := S_1 \cup \{[(A_0, \emptyset), r_1, (A_1, \phi_1), r_2, \dots, (A_{n-2}, \phi_{n-2}), r_{n-1}, (A_{n-1}, \phi_{n-1})]\}$
 - 7: $S_2 := \{[(B_0, \emptyset), s_1, \dots, s_m, (B_m, \psi_m), r, (B, \psi)] \mid ((B_m, \psi_m), r, (B, \psi)) \in E\}$
 - 8: **if** $n > 0 \wedge r = s_m^-$ **then**
 - 9: $S_2 := S_2 \cup \{[(B_0, \emptyset), s_1, (B_1, \psi_1), s_2, \dots, (B_{m-2}, \psi_{m-2}), s_{m-1}, (B_{m-1}, \psi_{m-1})]\}$
 - 10: $\text{result-concept} := \text{result-concept} \sqcap \prod_{\substack{l_1 \in S_1 \\ l_2 \in S_2}} \exists r.k\text{-lcs-r}(l_1, l_2, (V, E, S), k - 1)$
 - 11: **return** result-concept
-

p_1 extended by one more edge $(u_n, r, u) \in E$ or shorted by the last edge if $r_n = r^-$. In the first case soundness of completion for $(u_n, r, u) \in E$ yields $\text{vconcept}(u_n) \sqsubseteq_{\mathcal{T}} \exists r.\text{vconcept}(u)$ and thus $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \exists r.(\text{vconcept}(u) \sqcap \exists r^-. \text{lconcept}(p_1)) = \exists r.\text{lconcept}(l_1)$. In the second case we have $\text{lconcept}(p_1) = \text{vconcept}(u_n) \sqcap \exists r_n^-.\text{lconcept}(l_1) \sqsubseteq_{\mathcal{T}} \exists r.\text{lconcept}(l_1)$. Then the induction hypothesis yields that $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \exists r.k\text{-lcs-r}(l_1, l_2, (V, E, S), k - 1)$, therefore $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} L$ holds and by the same argument $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} L$ holds.

Minimality. To show that Algorithm 1 yields the *least* common subsumer w.r.t. the role-depth bound k , we show the following lemma.

Lemma 2. *Let p_1 and p_2 be two paths in the completion graph (V, E, S) with $p_1 = [u_0, r_1, \dots, r_n, u_n]$ and $p_2 = [v_0, s_1, \dots, s_m, v_m]$, such that $(u_{i-1}, r_i, u_i) \in E \setminus E_{\text{bad}}$ for all $1 \leq i \leq n$ and $(v_{j-1}, s_j, v_j) \in E \setminus E_{\text{bad}}$ for all $1 \leq j \leq m$, $u_0 = (A, \emptyset)$ and $v_0 = (B, \emptyset)$. Let $k \in \mathbb{N}$ and F an \mathcal{ELI} -concept description with $\text{rd}(F) \leq k$. If $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} F$ and $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} F$ then $L = k\text{-lcs-r}(p_1, p_2, (V, E, S), k) \sqsubseteq_{\mathcal{T}} F$.*

Proof. We prove this claim by induction on the role-depth bound k . For $k = 0$, $F = A_1 \sqcap \dots \sqcap A_n$ must be a conjunction of concept names. Since $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} F$

F and $lconcept(p_2) \sqsubseteq_{\mathcal{T}} F$, we have $lconcept(p_1) \sqsubseteq_{\mathcal{T}} A_i$ and $lconcept(p_2) \sqsubseteq_{\mathcal{T}} A_i$ for all $1 \leq i \leq n$. Since p_1 and p_2 only traverse edges over $E \setminus E_{bad}$, all possible rule applications of CI5 during that path were applied, and we have $vconcept(u_n) \sqsubseteq_{\mathcal{T}} A_i$ and $vconcept(v_m) \sqsubseteq_{\mathcal{T}} A_i$. Then completeness of the completion algorithm yields $A_i \in S(u_n)$ and $A_i \in S(v_m)$ for all $1 \leq i \leq n$. Thus, $L \sqsubseteq_{\mathcal{T}} F$.

For $k \geq 1$, F is a conjunction of concept names and existential restrictions. The concept names in F must appear in L by the same argument as in the base case. For each existential restriction $\exists r.F'$ of F , we can again use the fact that p_1 and p_2 only traverse edges over $E \setminus E_{bad}$ to derive that there must be nodes u and v with $vconcept(u) \sqsubseteq_{\mathcal{T}} F'$, $vconcept(v) \sqsubseteq_{\mathcal{T}} F'$, such that $vconcept(u_n) \sqsubseteq_{\mathcal{T}} \exists r.vconcept(u)$ and $vconcept(v_m) \sqsubseteq_{\mathcal{T}} \exists r.vconcept(v)$. Then completeness of completion yields that there are u' and v' with $(u_n, r, u') \in E \setminus E_{bad}$ or $u' = u_{n-1}, r = r_n^-$ and similarly $(v_m, r, v') \in E \setminus E_{bad}$ or $v' = v_{m-1}, r = s_m^-$, such that $vconcept(u') \sqsubseteq_{\mathcal{T}} vconcept(u)$ and $vconcept(v') \sqsubseteq_{\mathcal{T}} vconcept(v)$. Therefore, there are new paths $l_1 \in S_1$ and $l_2 \in S_2$, such that $lconcept(l_1) \sqsubseteq_{\mathcal{T}} F'$ and $lconcept(l_2) \sqsubseteq_{\mathcal{T}} F'$ which still only traverse edges in $E \setminus E_{bad}$, so the induction hypothesis yields $k\text{-lcs-r}(l_1, l_2, (V, E, S), k-1) \sqsubseteq_{\mathcal{T}} F'$, and thus $L = k\text{-lcs-r}(p_1, p_2, (V, E, S), k) \sqsubseteq_{\mathcal{T}} F$.

This shows that the Algorithm 1 computes the role-depth bounded least common subsumer for \mathcal{ELI} . In contrast to subsumption, the computation of $k\text{-lcs}$ does not increase complexity-wise when going from \mathcal{EL} to \mathcal{ELI} — it remains exponential in the size of k .

5 Conclusions and Future Work

In this paper we have extended the computation algorithm for the $k\text{-lcs}$ in \mathcal{EL} w.r.t. general TBoxes to two members of the \mathcal{EL} -family and showed that the proposed methods indeed compute the $k\text{-lcs}$. In cases where the exact lcs exists, our algorithms compute the exact lcs for a big enough k .

For \mathcal{ELI} the extension of the \mathcal{EL} algorithm for computing the $k\text{-lcs}$ required traversal of the completion graph w.r.t. paths and the correct handling of complex node labels.

In case of \mathcal{EL}^+ , the extension of the computation method for \mathcal{EL} turned out to be trivial, here our contribution rather lies in the simplification procedure devised. This procedure turned out to be extremely helpful, when reducing the concept size. For the NotGalen ontology the the result concepts were reduced by several orders of magnitude. It would be desirable to obtain the simplified \mathcal{EL}^+ -concept descriptions directly, instead of in the generate and then reduce kind of fashion employed so far. Besides this, we want to extend our results on \mathcal{EL}^+ and \mathcal{ELI} to the computation of most specific concepts by completion.

References

1. Turhan, A.-Y.: On the Computation of Common Subsumers in Description Logics. PhD thesis, TU Dresden, Institute for Theoretical Computer Science (2007)

2. d'Amato, C., Fanizzi, N., Esposito, F.: A dissimilarity measure for \mathcal{ALC} concept descriptions. In: Proceedings of the ACM symposium on Applied computing. SAC '06 (2006) 1695 – 1699
3. Janowicz, K.: Computing Semantic Similarity Among Geographic Feature Types Represented in Expressive Description Logics. PhD thesis, Institute for Geoinformatics, University of Münster, Germany (2008)
4. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. Journal of the American Medical Informatics Assoc. (2000) Fall Symposium Special Issue.
5. Consortium, T.G.O.: Gene Ontology: Tool for the unification of biology. Nature Genetics **25** (2000) 25–29
6. Rosse, C., Mejino, J.L.V.: A reference ontology for biomedical informatics: the foundational model of anatomy. Journal of Biomedical Informatics **36**(6) (2003) 478–500
7. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope further. In Clark, K., Patel-Schneider, P.F., eds.: In Proc. of the OWLED Workshop. (2008)
8. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Gottlob, G., Walsh, T., eds.: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03), Morgan Kaufmann (2003) 325–330
9. C. Lutz, R. Piro, and F. Wolter. Enriching \mathcal{EL} -concepts with greatest fixpoints. In Proc. of the 19th European Conf. on Artificial Intelligence (ECAI-10). IOS Press, (2010)
10. Peñaloza, R., Turhan, A.-Y.: A practical approach for computing generalization inferences in \mathcal{EL} . In Grobelnik, M., Simperl, E., eds.: Proc. of the 8th European Semantic Web Conf. (ESWC'11). Lecture Notes in Computer Science, Springer (2011)
11. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05, Edinburgh, UK, Morgan-Kaufmann Publishers (2005)
12. Vu, Q.H.: Subsumption in the description logic $\mathcal{EL}\mathcal{H}\mathcal{I}f_{R^+}$ w.r.t. general tboxes. Master's thesis, Technische Universität Dresden (2008)
13. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
14. A. Ecke and A.-Y. Turhan. Optimizations for the role-depth bounded least common subsumer in \mathcal{EL}^+ . In M. Horridge and P. Klinov, eds.: In Proc. of the OWLED Workshop, (2012) To appear.
15. Mendez, J., Ecke, A., Turhan, A.-Y.: Implementing completion-based inferences for the \mathcal{EL} -family. In Rosati, R., Rudolph, S., Zakharyashev, M., eds.: Proc. of the 2011 Description Logic Workshop (DL 2011). Volume 745., CEUR (2011)
16. Ecke, A.: Completion-based role-depth bounded least common subsumer for extensions of \mathcal{EL} . Belegarbeit, TU Dresden (2012) Available from <http://lat.inf.tu-dresden.de/~turhan/Teaching/AE-Beleg-12.pdf>.

Towards Practical Query Answering for Horn- \mathcal{SHIQ}^*

Thomas Eiter¹, Magdalena Ortiz¹, Mantas Šimkus¹
Trung-Kien Tran², and Guohui Xiao¹

¹ Institute of Information Systems, Vienna University of Technology
{eiter|ortiz|xiao}@kr.tuwien.ac.at
simkus@dbai.tuwien.ac.at

² STARLab, Vrije Universiteit Brussel
truntran@vub.ac.be

1 Introduction

Query answering has become a prominent reasoning task in Description Logics. This is witnessed not only by the high number of publications on the topic in the last decade, but also by the increasing number of query answering engines. A number of systems provide full conjunctive query (CQ) answering capabilities, including [1, 23, 25, 4, 11]. A common feature of these approaches is that they rely on existing technologies for relational or deductive databases. They focus on lightweight DLs like $\mathcal{DL-Lite}$ and \mathcal{EL} , and they use *query rewriting* to reduce the problem of answering a query over a DL ontology to a database query evaluation problem. For more expressive DLs that are not tractable in combined complexity, however, CQs (with the complete first-order semantics) are not yet supported by current reasoning engines. A range of algorithms has been designed, but they serve for theoretical purposes such as showing complexity bounds and are not amenable to practical implementation. The only exception is the rewriting algorithm implemented in the REQUIEM system, which covers \mathcal{ELHI} [23], an expressive extension of \mathcal{EL} for which standard reasoning is EXPTIME-hard.

In this paper, we contribute to the development of practical query answering systems beyond $\mathcal{DL-Lite}$ and \mathcal{EL} . We consider Horn- \mathcal{SHIQ} , the Horn fragment of the popular DL \mathcal{SHIQ} that underlies OWL DL. It combines all the expressive features of $\mathcal{DL-Lite}$ and \mathcal{EL} , and simultaneously extends them with transitive roles, qualified number restrictions and some universal quantification. Standard reasoning tasks in Horn- \mathcal{SHIQ} are already EXPTIME-hard in combined complexity but, due to the absence of disjunction, they are polynomial in data complexity. Since in Horn- \mathcal{SHIQ} models are significantly more complex than in $\mathcal{DL-Lite}$ and (most dialects of) \mathcal{EL} , extending existing query rewriting techniques is not straightforward.

The main contribution of this paper is a query answering method for Horn- \mathcal{SHIQ} that appears to be promising for practicable systems, as confirmed by the experimental evaluation of a prototype implementation.

– The core of the method is a novel query rewriting technique which transforms an input query q into a union Q of CQs (a UCQ) such that the answers of q over an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ with TBox \mathcal{T} and ABox \mathcal{A} coincide with the answers over \mathcal{A}

* This work was supported by the Austrian Science Fund (FWF) grants P20840 and T515.

of a Datalog program comprising Q and some rules to complete \mathcal{A} , showing Datalog-rewritability of CQ answering in Horn- \mathcal{SHIQ} .

– Naturally, the set Q may be exponential in the size of q in the worst case, but in practice we obtain rewritings Q that are of manageable size for real world ontologies. This is mostly due to the fact that new queries are only generated taking into account the anonymous domain elements implied by the terminology. Notably our algorithm is worst-case optimal in both data and combined complexity.

– We describe a prototype implementation of the approach that uses off-the-shelf Datalog reasoners. Despite being preliminary and lacking sophisticated optimizations, it shows that the approach is promising. It can answer queries efficiently over Horn- \mathcal{SHIQ} ontologies and scales down nicely to $\mathcal{DL-Lite}$, where it is competitive with state of the art query rewriting systems.

– The technique works for full Horn- \mathcal{SHIQ} and arbitrary CQs, but the implemented version does not support transitive (or more generally, non-simple) roles in the query. To keep presentation simple, we present here only the case without transitive roles, and refer to [6, 9] for a more general version with transitive roles and richer queries formulated in weakly DL-safe Datalog.

2 Preliminaries

Horn- \mathcal{SHIQ} The syntax and semantics of Horn- \mathcal{SHIQ} is defined in the usual way. A *role* is a role name p or its inverse p^- . A Horn- \mathcal{SHIQ} TBox \mathcal{T} in normal form is a set of *role inclusion axioms* $r \sqsubseteq s$, *transitivity axioms* $\text{trans}(r)$, and *general concept inclusion axioms (GCIs)* of the forms (F1) $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B$, (F2) $A_1 \sqsubseteq \forall r.B$, (F3) $A_1 \sqsubseteq \exists r.B$, and (F4) $A_1 \sqsubseteq \leq 1 r.B$, where A_1, \dots, A_n, B are concept names and r, s are roles. Axioms (F3) are called *existential*. W.l.o.g. we consider only Horn- \mathcal{SHIQ} TBoxes in normal form [16, 17]. We call s *transitive* in \mathcal{T} , if $\text{trans}(s) \in \mathcal{T}$ or $\text{trans}(s^-) \in \mathcal{T}$, and we call s *simple* in \mathcal{T} , if there is no transitive r in \mathcal{T} s.t. $r \sqsubseteq_{\mathcal{T}}^* s$, where \sqsubseteq^* is the reflexive transitive closure of $\{(r, s) \mid r \sqsubseteq s \in \mathcal{T} \text{ or } \text{inv}(r) \sqsubseteq \text{inv}(s) \in \mathcal{T}\}$. Only simple roles are allowed in axioms of the form (F4).

An *ABox* \mathcal{A} is a set of *assertions* $A(a)$ and $r(a, b)$, where A is a concept name, r a role, and a, b are individuals; the set of all individuals is denoted by N_I . An *ontology* is a pair $(\mathcal{T}, \mathcal{A})$ of a TBox \mathcal{T} and an ABox \mathcal{A} . The semantics is given by *interpretations* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ in the usual way.

A Horn- \mathcal{ALCHIQ} TBox is a Horn- \mathcal{SHIQ} TBox with no transitivity axioms. Horn- $\mathcal{ALCHIQ}^{\sqcap}$ TBoxes are obtained by allowing the *conjunction* $r_1 \sqcap r_2$ of roles r_1 and r_2 , interpreted $(r_1 \sqcap r_2)^{\mathcal{I}} = r_1^{\mathcal{I}} \cap r_2^{\mathcal{I}}$. We let $\text{inv}(r_1 \sqcap r_2) = \text{inv}(r_1) \sqcap \text{inv}(r_2)$ and assume w.l.o.g. that for each role inclusion $r \sqsubseteq s$ of a Horn- $\mathcal{ALCHIQ}^{\sqcap}$ TBox \mathcal{T} , (i) $\text{inv}(r) \sqsubseteq \text{inv}(s) \in \mathcal{T}$, and (ii) $s \in \{p, p^-\}$ for a role name p . For a set W and a concept or role conjunction $\Gamma = \gamma_1 \sqcap \dots \sqcap \gamma_m$, we write $\Gamma \subseteq W$ for $\{\gamma_1, \dots, \gamma_m\} \subseteq W$.

Conjunctive Queries A *conjunctive query (CQ)* is an expression of the form

$$q(\mathbf{u}) \leftarrow p_1(\mathbf{v}_1), \dots, p_m(\mathbf{v}_m)$$

where each $p_i(\mathbf{v}_i)$ is an *atom* of the form $A(x)$ or $r(x, y)$, where A is a concept name and r is a role, x, y are variables, and q is a special *query predicate* not occurring elsewhere. For convenience, we may identify such a CQ with the set of its atoms, and

Table 1: Inference rules. M^θ, N^θ , (resp., S^θ) are conjunctions of atomic concepts (roles); A, B are atomic concepts

$\frac{M \sqsubseteq \exists S.(N \sqcap N') \quad N \sqsubseteq A}{M \sqsubseteq \exists S.(N \sqcap N' \sqcap A)} \mathbf{R}_{\sqsubseteq}^{\exists}$	$\frac{M \sqsubseteq \exists(S \sqcap S').N \quad S \sqsubseteq r}{M \sqsubseteq \exists(S \sqcap S' \sqcap r).N} \mathbf{R}_{\sqsubseteq}^r$	$\frac{M \sqsubseteq \exists S.(N \sqcap \perp)}{M \sqsubseteq \perp} \mathbf{R}_{\perp}$
$\frac{M \sqsubseteq \exists(S \sqcap r).N \quad A \sqsubseteq \forall r.B}{M \sqcap A \sqsubseteq \exists(S \sqcap r).(N \sqcap B)} \mathbf{R}_{\forall}$	$\frac{M \sqsubseteq \exists(S \sqcap \text{inv}(r)).(N \sqcap A) \quad A \sqsubseteq \forall r.B}{M \sqsubseteq B} \mathbf{R}_{\forall}^{-}$	
$\frac{M \sqsubseteq \exists(S \sqcap r).(N \sqcap B) \quad A \sqsubseteq \leq 1 r.B \quad M' \sqsubseteq \exists(S' \sqcap r).(N' \sqcap B)}{M \sqcap M' \sqcap A \sqsubseteq \exists(S \sqcap S' \sqcap r).(N \sqcap N' \sqcap B)} \mathbf{R}_{\leq}$		
$\frac{M \sqsubseteq \exists(S \sqcap \text{inv}(r)).(N_1 \sqcap N_2 \sqcap A) \quad A \sqsubseteq \leq 1 r.B \quad N_1 \sqcap A \sqsubseteq \exists(S' \sqcap r).(N' \sqcap B \sqcap C)}{M \sqcap B \sqsubseteq C \quad M \sqcap B \sqsubseteq \exists(S \sqcap \text{inv}(S' \sqcap r)).(N_1 \sqcap N_2 \sqcap A)} \mathbf{R}_{\leq}^{-}$		

use $q(\mathbf{u})$ (or simply q) to refer to it. We call $\mathbf{u} \subseteq \bigcup_{1 \leq i \leq m} \mathbf{v}_i$ the *distinguished* variables of q . A *match* for q in \mathcal{I} is a mapping from variables in q to elements in $\Delta^{\mathcal{I}}$ such that $\pi(\mathbf{t}) \in p^{\mathcal{I}}$ for each atom $p(\mathbf{t})$ of q . The *answer* to q over \mathcal{O} is the set of all $\mathbf{c} \in \mathbb{N}_1^{|\mathbf{u}|}$ such that in every model \mathcal{I} of \mathcal{O} some match π for q exists with $\pi(\mathbf{u}) = (\mathbf{c})^{\mathcal{I}}$.

Elimination of Transitivity. As usual, transitivity axioms roles can be eliminated from Horn- \mathcal{SHIQ} TBoxes. To obtain a Horn- \mathcal{ALCHIQ} TBox \mathcal{T}^* that is also in normal form, we can use the transformation from [16]. This transformation preserves satisfiability and, provided that queries contain only simple roles, also query answers (answers are not preserved for arbitrary queries unless the notion of match is suitably relaxed). In the rest of the paper we describe a procedure for answering CQs in Horn- $\mathcal{ALCHIQ}^{\square}$.

3 Canonical Models

For answering CQs in Horn DLs usually the *canonical model property* is employed [7, 21, 2]. In particular, for a consistent Horn- $\mathcal{ALCHIQ}^{\square}$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, there exists a model \mathcal{I} of \mathcal{O} that can be homomorphically mapped into any other model \mathcal{I}' of \mathcal{O} . We show that such an \mathcal{I} can be built in three steps:

- (1) close \mathcal{T} under specially tailored inferences rules,
- (2) close \mathcal{A} under all but existential axioms of \mathcal{T} , and
- (3) extend \mathcal{A} by “applying” the existential axioms of \mathcal{T} .

For Step (1) we use the calculus in Table 1, which is similar to [16, 20]. Given a Horn- $\mathcal{ALCHIQ}^{\square}$ TBox \mathcal{T} , we denote by $\Xi(\mathcal{T})$ the TBox obtained from \mathcal{T} by exhaustively applying the inference rules in Table 1. In Step (2) we simply ‘apply’ in the ABox all but existential axioms in \mathcal{T} . For convenience, this is done using the set $\text{cr}(\mathcal{T})$ of Datalog rules in Table 2. Since every ABox \mathcal{A} can be seen as a set of Datalog facts, $\mathcal{A} \cup \text{cr}(\mathcal{T})$ is a Datalog program (with constraints) which has a unique minimal Herbrand model $\mathcal{J} = \text{MM}(\mathcal{A} \cup \text{cr}(\mathcal{T}))$ if \mathcal{O} is consistent. This model is almost a canonical model of $(\mathcal{T}, \mathcal{A})$; however, existential axioms may be violated. To deal with this, in Step (3) we extend \mathcal{J} with new domain elements as required by axioms $M \sqsubseteq \exists r.N$ in $\Xi(\mathcal{T})$, using a procedure similar to the well known *chase* in databases.

Table 2: (Completion rules) Datalog program $\text{cr}(\mathcal{T})$.

$B(y) \leftarrow A(x), r(x, y)$ for each $A \sqsubseteq \forall r. B \in \mathcal{T}$ $B(x) \leftarrow A_1(x), \dots, A_n(x)$ for all $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B \in \Xi(\mathcal{T})$ $r(x, y) \leftarrow r_1(x, y), \dots, r_n(x, y)$ for all $r_1 \sqcap \dots \sqcap r_n \sqsubseteq r \in \mathcal{T}$ $\perp(x) \leftarrow A(x), r(x, y_1), r(x, y_2), B(y_1), B(y_2), y_1 \neq y_2$ for each $A \sqsubseteq \leq 1 r. B \in \mathcal{T}$ $\Gamma \leftarrow A(x), A_1(x), \dots, A_n(x), r(x, y), B(y)$ for all $A_1 \sqcap \dots \sqcap A_n \sqsubseteq \exists (r_1 \sqcap \dots \sqcap r_m). (B_1 \sqcap \dots \sqcap B_k)$ and $A \sqsubseteq \leq 1 r. B$ of $\Xi(\mathcal{T})$ such that $r=r_i$ and $B=B_j$ for some i, j with $\Gamma \in \{B_1(y), \dots, B_k(y), r_1(x, y), \dots, r_k(x, y)\}$

Definition 1. Let \mathcal{T} be a Horn- \mathcal{ALCHIQ}^\sqcap TBox and let \mathcal{I} be an interpretation. A GCI $M \sqsubseteq \exists S.N$ is applicable at $e \in \Delta^\mathcal{I}$ if (a) $e \in M^\mathcal{I}$, (b) there is no $e' \in \Delta^\mathcal{I}$ with $(e, e') \in S^\mathcal{I}$ and $e' \in N^\mathcal{I}$, (c) there is no axiom $M' \sqsubseteq \exists S'.N' \in \mathcal{T}$ such that $e \in (M')^\mathcal{I}$, $S \subseteq S'$, $N \subseteq N'$, and $S \subset S'$ or $N \subset N'$. An interpretation \mathcal{J} obtained from \mathcal{I} by an application of an applicable axiom $M \sqsubseteq \exists S.N$ at $e \in \Delta^\mathcal{I}$ is defined as:

- $\Delta^\mathcal{J} = \Delta^\mathcal{I} \cup \{d\}$ with d a new element not present in $\Delta^\mathcal{I}$ (we call d a successor of e),
- For each concept name A and each $o \in \Delta^\mathcal{J}$, we have $o \in A^\mathcal{J}$ if (a) $o \in \Delta^\mathcal{I}$ and $o \in A^\mathcal{I}$; or (b) $o = d$ and $A \in N$.
- For each role name r and $o, o' \in \Delta^\mathcal{J}$, we have $(o, o') \in r^\mathcal{J}$ if (a) $o, o' \in \Delta^\mathcal{I}$ and $(o, o') \in r^\mathcal{I}$; or (b) $(o, o') = (e, d)$ and $r \in S$; or (c) $(o, o') = (d, e)$ and $\text{inv}(r) \in S$.

We denote by $\text{chase}(\mathcal{I}, \mathcal{T})$ a possibly infinite interpretation obtained from \mathcal{I} by applying the existential axioms in \mathcal{T} . We require fairness: the application of an applicable axiom can not be infinitely postponed.

We note that $\text{chase}(\mathcal{I}, \mathcal{T})$ is unique up to renaming of domain elements. As usual in DLs, it can be seen as a ‘forest’: the application of existential axioms simply attaches ‘trees’ to an arbitrarily shaped model \mathcal{I} .

Theorem 1. Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be a Horn- \mathcal{ALCHIQ}^\sqcap ontology. Then \mathcal{O} is consistent iff $\mathcal{A} \cup \text{cr}(\mathcal{T})$ consistent. Moreover, if \mathcal{O} is consistent, then (a) $\text{chase}(\text{MM}(\mathcal{A} \cup \text{cr}(\mathcal{T})), \Xi(\mathcal{T}))$ is a model of \mathcal{O} , and (b) $\text{chase}(\text{MM}(\mathcal{A} \cup \text{cr}(\mathcal{T})), \Xi(\mathcal{T}))$ can be homomorphically mapped into any model of \mathcal{O} .

The proof of Theorem 1 can be found in [9]; see [21] for a proof of a similar result.

Observe that checking consistency of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ reduces to evaluating the Datalog program $\mathcal{A} \cup \text{cr}(\mathcal{T})$. We note that $\Xi(\mathcal{T})$ can be computed in exponential time in the size of \mathcal{T} : the calculus only infers axioms of the form $M \sqsubseteq B$ and $M \sqsubseteq \exists S.N$, where M, N are conjunctions of atomic concepts, B is atomic and S is a conjunction of roles, and there are exponentially many such axioms.

4 Query Rewriting

The following theorem, which is immediate from Theorem 1, allows us to concentrate on models obtained by the chase procedure.

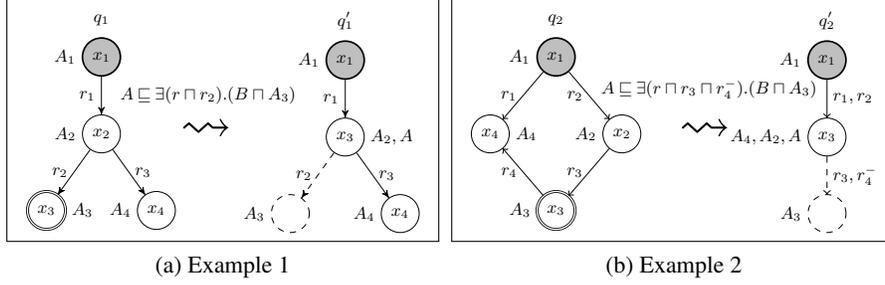


Fig. 1: Examples of query rewriting

Theorem 2. Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be a Horn- \mathcal{ALCHIQ}^\square ontology. Then $\mathcal{A} \cup \text{cr}(\mathcal{T})$ is consistent iff \mathcal{O} is consistent. Moreover, if \mathcal{O} is consistent, then $\text{ans}(\mathcal{O}, q) = \text{ans}(\mathcal{I}_{\mathcal{O}}, q)$, where $\mathcal{I}_{\mathcal{O}} = \text{chase}(\text{MM}(\mathcal{A} \cup \text{cr}(\mathcal{T})), \Xi(\mathcal{T}))$.

Computing $\text{ans}(\mathcal{I}_{\mathcal{O}}, q)$ is still not trivial, as $\mathcal{I}_{\mathcal{O}}$ can be infinite. Hence, we rewrite q into a set Q of CQs such that $\text{ans}(\mathcal{I}_{\mathcal{O}}, q) = \bigcup_{q' \in Q} \text{ans}(\text{MM}(\mathcal{A} \cup \text{cr}(\mathcal{T})), q')$, that is, we can evaluate them over the finite $\text{MM}(\mathcal{A} \cup \text{cr}(\mathcal{T}))$.

The intuition is the following. Suppose q has a non-distinguished variable x , and that there is some match π in $\mathcal{I}_{\mathcal{O}}$ such that $\pi(x)$ is an object in the ‘tree part’ introduced by the chase procedure, and it has no descendant in the image of π . Then for all atoms $r(y, x)$ of q , the ‘neighbor’ variable y must be mapped to the parent of $\pi(x)$. A rewrite step makes a choice of such an x , and employs an existential axiom from $\Xi(\mathcal{T})$ to ‘clip off’ x , eliminating all query atoms involving it. By repeating this procedure, we can clip off all variables matched in the tree part and obtain a query with a match in $\text{MM}(\mathcal{A} \cup \text{cr}(\mathcal{T}))$.

Definition 2 (Query rewriting). For a CQ q and a Horn- \mathcal{ALCHIQ}^\square TBox \mathcal{T} , we write $q \rightarrow_{\mathcal{T}} q'$ if q' can be obtained from q in the following steps:

- (S1) Select in q an arbitrary non-distinguished variable x such that there are no atoms of the form $r(x, x)$ in q .
- (S2) Replace each role atom $r(x, y)$ in q , where y is arbitrary, by the atom $\text{inv}(r)(y, x)$.
- (S3) Let $V_p = \{y \mid \exists r : r(y, x) \in q\}$, and select some $M \sqsubseteq \exists S.N \in \Xi(\mathcal{T})$ such that
 - (a) $\{r \mid r(y, x) \in q \wedge y \in V_p\} \subseteq S$, and
 - (b) $\{A \mid A(x) \in q\} \subseteq N$.
- (S4) Drop from q each atom containing x .
- (S5) Rename each $y \in V_p$ of q by x .
- (S6) Add the atoms $\{A(x) \mid A \in M\}$ to q .

We write $q \rightarrow_{\mathcal{T}}^* q'$ if $q = q_0$ and $q' = q_n$ for some finite rewrite sequence $q_0 \rightarrow_{\mathcal{T}} q_1 \cdots \rightarrow_{\mathcal{T}} q_n$, $n \geq 0$. Furthermore, we let $\text{rew}_{\mathcal{T}}(q) = \{q' \mid q \rightarrow_{\mathcal{T}}^* q'\}$.

Example 1. The query $q_1(x_1) \leftarrow A_1(x_1), r_1(x_1, x_2), A_2(x_2), r_2(x_2, x_3), A_3(x_3), r_3(x_2, x_4), A_4(x_4)$ is depicted on the left hand side of Figure 1a. The node in bold corresponds to the answer variable x_1 . Assume that $A \sqsubseteq \exists(r \sqcap r_2).(B \sqcap A_3)$ and $A \sqsubseteq \exists(r \sqcap r_3 \sqcap r_4).(B$

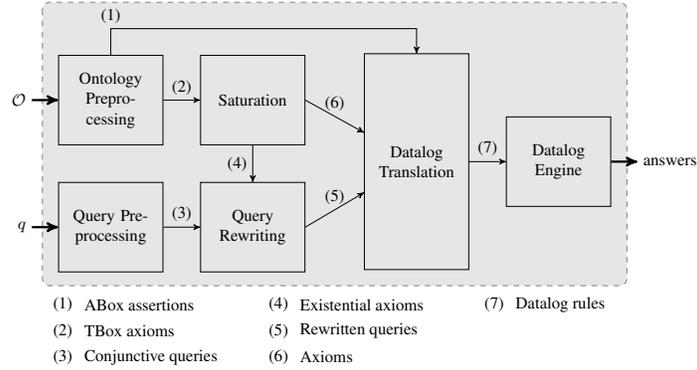


Fig. 2: CLIPPER system architecture

$\sqcap A_3$) are in $\Xi(\mathcal{T})$. If we pick for (S1) the variable x_3 , we get $V_p = \{x_2\}$ and we can select $A \sqsubseteq \exists(r \sqcap r_2).(B \sqcap A_3) \in \Xi(\mathcal{T})$, as it satisfies (S3.a) and (S3.b). After performing (S4), (S5) and (S6) we obtain the rewritten query $q'_1(x_1) \leftarrow A_1(x_1), r_1(x_1, x_3), A_2(x_3), A(x_3), r_3(x_3, x_4), A_4(x_4)$. Intuitively, we can safely remove from q_1 all atoms containing x_3 because the added atom $A(x_3)$ ensures that whenever q'_1 has a match so does q_1 .

Example 2 (ctd). Now we consider the query $q_2(x_1) \leftarrow A_1(x_1), r_2(x_1, x_2), A_2(x_2), r_3(x_2, x_3), A_3(x_3), r_1(x_1, x_4), A_4(x_4), r_4(x_3, x_4)$ in Figure 1b. We choose the variable x_3 , replace $r_4(x_3, x_4)$ by $r_4^-(x_4, x_3)$ in step (S2), and get $V_p = \{x_2, x_4\}$. Intuitively, if $\pi(x_3)$ is a leaf in a tree-shaped match π , then x_2 and x_4 must both be mapped to the parent of $\pi(x_3)$. Since the GCI $A \sqsubseteq \exists(r \sqcap r_3 \sqcap r_4^-).(B \sqcap A_3)$ in $\Xi(\mathcal{T})$ satisfies (S3.a,b), we can drop the atoms containing x_3 from q_2 , and perform (S5) and (S6) to obtain the rewritten query $q'_2(x_1) \leftarrow A_1(x_1), r_1(x_1, x_3), r_2(x_1, x_3), A_4(x_3), A_2(x_3), A(x_3)$.

Now we can state our main result (see [9] for the proof of a more general result):

Theorem 3. *Suppose $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is a consistent Horn- \mathcal{ALCHIQ}^\square ontology and let q be a CQ. Then $ans(\mathcal{O}, q) = \bigcup_{q' \in \text{rew}_{\mathcal{T}}(q)} ans(MM(\mathcal{A} \cup \text{cr}(\mathcal{T})), q')$.*

By the above reduction, we can answer q over $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ by evaluating $\text{rew}_{\mathcal{T}}(q)$ over $MM(\mathcal{A} \cup \text{cr}(\mathcal{T}))$ or, equivalently, by evaluating the Datalog program $\text{rew}_{\mathcal{T}}(q) \cup \mathcal{A} \cup \text{cr}(\mathcal{T})$ and collecting the tuples \mathbf{u} with $q(\mathbf{u})$ in the minimal model. We note that $\text{rew}_{\mathcal{T}}(q)$ is finite and computable in time exponential in the size of \mathcal{T} and q : rules in $\text{rew}_{\mathcal{T}}(q)$ use only relation names and variables that occur in q and \mathcal{T} . Furthermore, the *grounding* of $\text{rew}_{\mathcal{T}}(q) \cup \mathcal{A} \cup \text{cr}(\mathcal{T})$ is exponential in the size of \mathcal{O} , but polynomial for fixed \mathcal{T} and q . By the complexity of Datalog, it follows that the resulting algorithm is exponential in combined but polynomial in data complexity; this is worst-case optimal [7].

5 Implementation

To evaluate the feasibility of the new rewriting, we have implemented a prototype system CLIPPER,³ which supports CQ answering over Horn- \mathcal{SHIQ} ontologies (non-simple

³ <http://www.kr.tuwien.ac.at/research/systems/clipper/>

Algorithm 1: Answering CQs via Query Rewriting

Input: Horn-*SHIQ* ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, Conjunctive Query q
Output: query results

```
 $\mathcal{T} \leftarrow \text{Normalize}(\mathcal{T});$  ▷ Normalization  
 $\mathcal{T}^* \leftarrow \text{ElimTrans}(\mathcal{T});$  ▷ Eliminate Transitive Roles  
 $\Xi(\mathcal{T}^*) \leftarrow \text{Saturate}(\mathcal{T}^*);$  ▷ TBox Saturation  
 $Q \leftarrow \text{Rewrite}(q, \Xi(\mathcal{T}^*));$  ▷ Query Rewriting  
 $\text{cr}(\mathcal{T}) \leftarrow \text{CompletionRules}(\mathcal{T});$  ▷ Completion Rules  
 $\mathcal{P} = \mathcal{A} \cup \text{cr}(\mathcal{T}) \cup Q;$  ▷ Datalog Translation  
 $\text{ans} \leftarrow \{\mathbf{u} \mid q(\mathbf{u}) \in \text{MinimalModel}(\mathcal{P})\};$  ▷ Call Datalog Reasoner  
return  $\text{ans};$ 
```

roles are disallowed in queries). To the best of our knowledge, it is the first such system for Horn-*SHIQ* (under the standard semantics of first-order logic), and in expressiveness subsumes similar *DL-Lite* and *EL* reasoning engines (see below).

We describe the architecture of CLIPPER in Figure 2, and the main steps in Algorithm 1. CLIPPER is implemented in Java and uses OWLAPI 3.2.2 [13] for parsing ontologies. It accepts an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ and a CQ q in the SPARQL syntax as input. For efficiency reasons we implemented a lightweight ontology representation: all concepts, roles and individuals are encoded as integers; the conjunction of concepts and roles are stored in hash sets. Since we often need to manipulate large tables of axioms, we built inverted indexes over such axioms to support fast lookup and matching.

Ontology Preprocessing. This component is responsible for (1) ontology parsing (using OWLAPI 3.2.2), (2) profile checking and ontology normalization [16], and (3) converting the ontology into the internal format.

Query Preprocessing. This component simply parses CQs in SPARQL syntax and converts them into the internal format.

Saturation. This component exhaustively applies the saturation rules in Table 1 on TBox. We use the index structure to find which rules can be applied and the new axioms are generated incrementally.

Query Rewriting. This component uses Algorithm 2 to rewrite the input q . It implements the rewriting step from Definition 2, exhaustively traversing all existential axioms in $\Xi(\mathcal{T})$ for all the non-distinguished variables. The index structure helps the system efficiently search through the set of existential axioms while rewriting.

Datalog Translation. This component generates a Datalog program with the rewritten set of queries Q , the completion rules $\text{cr}(\mathcal{T})$ in Table 2, and the facts in \mathcal{A} .

Datalog Engine. The resulting program is evaluated using the Datalog engine DLV-20101014 [18] or Clingo 3.0.3 [10]. If the program (and hence the ontology) is consistent, its minimal model is returned and the answer tuples are filtered from it.

6 Experiments

We tested CLIPPER on a Pentium Core2 Duo 2.00GHZ with 2GB RAM under Ubuntu 10.04 and 512MB heap for the Java VM. We conducted the following experiments.

Algorithm 2: Rewrite(q, \mathcal{T})

Input: CQ q with only simple roles; TBox \mathcal{T}
Output: Rewritten queries of q w.r.t. \mathcal{T}
 $\text{rew}_{\mathcal{T}}(q) \leftarrow \emptyset;$ \triangleright will be updated from the sub procedure
 $\text{rewrite}(q);$ \triangleright Call sub procedure
return $\text{rew}_{\mathcal{T}}(q);$

Sub Procedure $\text{rewrite}(q)$
 $\text{rew}_{\mathcal{T}}(q) \leftarrow \text{rew}_{\mathcal{T}}(q) \cup \{q\};$
foreach *non-distinguished variables x of q* **do**
 if $r(x, x) \notin q$ **then**
 Replace each $r(x, y)$ in q by $r^-(y, x);$
 $S \leftarrow \{r \mid r(y, x) \in q\};$ $P \leftarrow \{y \mid r(y, x) \in q\};$ $N \leftarrow \{A \mid A(x) \in q\};$
 foreach $M \sqsubseteq \exists S' N' \in \mathcal{T}$ **do**
 if $S \subseteq S'$ **and** $N \subseteq N'$ **then**
 Obtain q' from q by:
 begin
 (1) Drop from q each atom containing the variable $x;$
 (2) Rename each $y \in P$ by $x;$
 (3) Add $\{A(x) \mid A \in M\}$ to $q;$
 if $q' \notin \text{rew}_{\mathcal{T}}(q)$ **then**
 $\text{rewrite}(q');$ \triangleright Recursion
 end

1. Downscaling test. We compared CLIPPER with other query rewriting systems for *DL-Lite*, viz. REQUIEM (Perez-Urbina et al. [23]) and PRESTO [25], and found that it is competitive and scales down well on *DL-Lite* ontologies. We used the ontologies and queries (Q1–Q5) from the REQUIEM test suite, which have been widely used for system tests; in addition we considered the queries in Table 3a.

Table 3b shows the number of rewritten queries and the rewriting time for the ontologies ADOLENA (A), STOCK-EXCHANGE (S), VICODI (V), and UNIVERSITY (U); the rewriting time excludes loading and preprocessing. CLIPPER and PRESTO generated in most cases rule sets of comparable size, and in short time. In a few cases PRESTO generated significantly less rules than CLIPPER, and only for V PRESTO was notably faster. REQUIEM generated in several cases significantly more rules, despite considering the G-version which generates optimized rules (and hence uses considerably more time). The difference seems to be caused by rule unfolding required in their rewriting.

For UNIVERSITY, the only ontology in the suite having an ABox, we evaluated the rewritten queries over four different ABoxes (67k to 320k assertions) using DLV. Interestingly, in all cases the execution times for the three rewritings were very similar; the average runtime of each query on the four ABoxes is shown in brackets.

2. Full Horn-*SHIQ*. To test CLIPPER on a full Horn-*SHIQ* ontology, we modified the UOBM ontology [19], which is in *SHOIN(D)*, by dropping or strengthening (in case of disjunctions) non-Horn-*SHIQ* TBox axioms; the final ontology has 196 TBox axioms. We used ABoxes \mathcal{A}_i , $1 \leq i \leq 4$, with 20k, 80k, 140k and 200k assertions. The test queries in Table 4a were tailored to require reasoning with Horn-*SHIQ* constructs

Table 3: Experiments with \mathcal{DL} -Lite ontology
(a) Additional Queries

A	Q6(x) \leftarrow Device(x), assistsWith(x,y), ReadingDevice(y) Q7(x) \leftarrow Device(x), assistsWith(x,y), ReadingDevice(y), assistsWith(y,z), SpeechAbility(z)
S	Q6(x,z) \leftarrow Investor(x), hasStock(x,y), Stock(y), Company(z), hasStock(z,y) Q7(x,z,w) \leftarrow Investor(x), hasStock(x,y), Stock(y), isListedIn(y,z), StockExchangeList(z), Company(w), hasStock(w,y)
U	Q6(x,y) \leftarrow Professor(x), teacherOf(x,y), GraduateCourse(y) Q7(x,z) \leftarrow Faculty(y), Professor(z), Student(x), memberOf(x,y), worksFor(z,y)
V	Q6(x,y,z) \leftarrow Person(x), hasRole(x,y), Leader(y), exists(y,z) Q7(x,y,z,w) \leftarrow Person(x), hasRole(x,y), Leader(y), exists(y,z), TemporalInterval(z), related(x,w), Country(w)

(b) Downscaling evaluation

		# Rules/CQs			Time (ms)					# Rules/CQs			Time (ms)		
		RG	Presto	CLIPPER	RG	Presto	CLIPPER			RG	Presto	CLIPPER	RG	Presto	CLIPPER
A	Q1	27	53	42	281	45	50	Q1	15	16	15	13	8	73	
	Q2	50	32	31	184	46	62	Q2	10	3	10	16	10	58	
	Q3	104	32	31	292	27	65	Q3	72	28	26	77	12	63	
	Q4	224	43	36	523	32	71	V Q4	185	44	41	261	17	71	
	Q5	624	37	36	1177	25	70	Q5	30	16	8	99	15	44	
	Q6	364	35	30	523	31	65	Q6	18	22	18	27	11	69	
	Q7	2548	43	32	7741	61	64	Q7	180	34	27	359	12	105	
S	Q1	6	7	10	14	7	19	Q1	2	4	2	14 (1247)	12 (1252)	27 (1255)	
	Q2	2	3	22	263	9	22	Q2	1	2	45	201 (1247)	23 (1262)	36 (1637)	
	Q3	4	4	9	1717	10	21	Q3	4	8	17	477 (2055)	26 (2172)	29 (1890)	
	Q4	4	4	24	1611	9	23	U Q4	2	56	63	2431 (1260)	20 (1235)	28 (1735)	
	Q5	8	5	10	18941	10	22	Q5	10	8	16	7216 (1267)	26 (1305)	36 (1372)	
	Q6	4	8	5	204	11	21	Q6	10	13	10	13 (1272)	14 (1260)	27 (1262)	
	Q7	8	6	7	1733	11	17	Q7	960	24	19	1890 (1730)	15 (1310)	35 (1322)	

unavailable in \mathcal{DL} -Lite and \mathcal{EL} . Table 4b shows the number of rewritten queries, rewriting time and DLV running time. We see that CLIPPER answered all queries in reasonable time and scaled well (time printed $\mathcal{A}_1 / \mathcal{A}_2 / \mathcal{A}_3 / \mathcal{A}_4$). The rewriting times for all the queries are small and at most within a factor of 3. The high number of rules generated for Q3 is due to many different possibilities for deriving some atoms in the query, like Person(x). However, the evaluation still performs well (it stays within a small factor).

7 Related Work

Since Calvanese et al. introduced query rewriting in their seminal work on \mathcal{DL} -Lite [3], many query rewriting techniques have been developed and implemented, e.g. (Perez-Urbina et al. [23], Rosati and Almatelli [25], Chortaras et al. [4], Gottlob et al. [11]), usually aiming at an optimized rewriting size. Some of them also go beyond \mathcal{DL} -Lite; e.g. Perez-Urbina et al. cover \mathcal{ELHI} , while Gottlob et al. consider Datalog^\pm . Most approaches rewrite a query into a (union of) CQs; in [25] a non-recursive Datalog program is generated, while Perez-Urbina et al. produce a CQ for \mathcal{DL} -Lite and a (recursive) Datalog program for DLs of the \mathcal{EL} family. Our approach rewrites a CQ into a union of CQs, but generates possibly recursive Datalog rules to capture the TBox. The closest DL to Horn- \mathcal{SHIQ} for which a rewriting technique has been implemented is \mathcal{ELHI} [23]. Unlike \mathcal{ELHI} , Horn- \mathcal{SHIQ} can express functionality, a feature supported by \mathcal{DL} -Lite

Table 4: Experiments with UOBM Horn- \mathcal{SHIQ} ontology

(a) Queries	(b) Running time			
	# of Rules	Rew (ms)	Datalog time (DLV) (ms)	
Q1(x) \leftarrow worksFor(x, y), isAffiliatedOrganizationOf(y, z), College(z)	3	87	120/ 370/ 620/ 870	
Q2(x) \leftarrow Postdoc(x), worksFor(x, y), University(y), hasAlumnus(y, x)	16	98	130/ 380/ 630/ 880	
Q3(x) \leftarrow Person(x), like(x, y), Chair(z), isHeadOf(z, w), like(z, y)	180	212	370/ 890/ 1420/ 1960	
Q4(x) \leftarrow takeCourse(x, y), GraduateCourse(y), isTaughtBy(y, z), Professor(z)	45	156	180/ 480/ 780/ 1070	
Q5(x, z) \leftarrow LeisureStudent(x), takesCourse(x, y), CSCourse(y), isStudentOf(x, z), University(z)	37	152	160/ 440/ 720/ 1010	
Q6(x, y) \leftarrow enrollIn(x, y), hasDegreeFrom(x, y), University(y)	16	112	130/ 370/ 630/ 880	
Q7(x, y) \leftarrow PeopleWithManyHobbies(x), isMemberOf(x, z), like(x, w), TennisClass(w), hasMember(z, y), like(y, w)	55	143	180/ 470/ 750/ 1050	
Q8(x, z) \leftarrow TennisFan(x), like(x, y), Sport(y), isHeadOf(x, z), ReserachGroup(z)	17	105	130/ 380/ 630/ 870	
Q9(x, y, z) \leftarrow Student(x), hasDegreeFrom(x, y), Professor(z), worksFor(z, y), isAdvisorOf(z, x)	33	126	150/ 430/ 720/ 1010	
Q10(x, y, w) \leftarrow Professor(x), Dean(y), isMemberOf(y, w), worksFor(x, w), hasPublication(x, z), isPublicationOf(z, y)	23	137	150/ 390/ 640/ 880	

considered relevant for applications. A comparison of both systems on ontologies beyond $\mathcal{DL-Lite}$ remains for future work. Our technique resembles Rosati’s for CQs in \mathcal{EL} [24], which incorporates the CQ into the TBox *before* saturation and then (after saturation) translates it into Datalog, resulting in a best-case exponential algorithm. We avoid this by doing a rewrite step only if the TBox has an applicable existential axiom.

Rewriting approaches for more expressive DLs are less common. The most notable exception is Hustadt et al.’s translation of \mathcal{SHIQ} terminologies into disjunctive Datalog [15], which is implemented in the KAON2 reasoner. The latter can be used to answer queries over arbitrary ABoxes, but supports only instance queries. An extension to CQs (without transitive roles) is given in [14], but it is not implemented. To our knowledge, also the extension of the rewriting in [23] to nominals remains to be implemented [22]. In [20] a Datalog rewriting is used to establish complexity bounds of standard reasoning in the Horn fragments of \mathcal{SHOIQ} and \mathcal{SROIQ} , but it does not cover CQs.

8 Conclusion

We presented a rewriting-based algorithm for answering CQs over Horn- \mathcal{SHIQ} ontologies. Our prototype implementation shows potential for practical applications, and further optimizations will improve it. Future versions of CLIPPER will support transitive roles and queries formulated in weakly DL-safe Datalog, for which the theoretic foundations have been already developed here and in [9].

As an interesting application, we mention that our method allows to improve reasoning with \mathcal{DL} -programs, which loosely couple rules and ontologies [5]. To avoid the overhead caused by the interaction of a rule reasoner and an ontology reasoner of traditional methods, the *inline evaluation* framework translates ontologies into rules [12, 8]. The techniques of this paper can be faithfully integrated into the inline evaluation framework to efficiently evaluate DL-programs involving Horn- \mathcal{SHIQ} ontologies.

References

1. Acciari, A., Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. In: AAI. pp. 1670–1671 (2005)

2. Cali, A., Gottlob, G., Lukasiewicz, T.: Datalog[±]: a unified approach to ontologies and integrity constraints. In: ICDT'09. pp. 14–30. ACM (2009)
3. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning* 39(3), 385–429 (2007)
4. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting for OWL 2 QL. In: CADE'11. pp. 192–206. Springer-Verlag (2011)
5. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence* 172(12–13), 1495–1539 (2008)
6. Eiter, T., Ortiz, M., Šimkus, M., Tran, T., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: AAI'12 (2012), (To appear)
7. Eiter, T., Gottlob, G., Ortiz, M., Simkus, M.: Query answering in the description logic Horn-SHIQ. In: JELIA'08. pp. 166–179. Springer (2008)
8. Eiter, T., Krennwallner, T., Schneider, P., Xiao, G.: Uniform evaluation of nonmonotonic DL-programs. In: FoKS'12. LNCS, vol. 7153, pp. 1–22. Springer (March 2012)
9. Eiter, T., Ortiz, M., Šimkus, M., Tran, T., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. Tech. Rep. INFSYS RR-1843-12-04, TU Vienna (2012), <http://www.kr.tuwien.ac.at/research/reports/rr1204.pdf>
10. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.T.: Potassco: The potsdam answer set solving collection. *AI Commun.* 24(2), 107–124 (2011)
11. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: ICDE'11. pp. 2–13 (2011)
12. Heymans, S., Eiter, T., Xiao, G.: Tractable reasoning with DL-programs over Datalog-rewritable description logics. In: ECAI'10. pp. 35–40. IOS Press (2010)
13. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. *Semantic Web* 2(1), 11–21 (2011)
14. Hustadt, U., Motik, B., Sattler, U.: A decomposition rule for decision procedures by resolution-based calculi. In: LPAR'04. pp. 21–35. Springer (2004)
15. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive Datalog. *J. Autom. Reasoning* 39(3), 351–384 (2007)
16. Kazakov, Y.: Consequence-driven reasoning for Horn SHIQ ontologies. In: IJCAI'09. pp. 2040–2045 (2009)
17. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for Horn description logics. In: AAI'07. pp. 452–457. AAI Press (2007)
18. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. *ACM ToCL* 7(3), 499–562 (2006)
19. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: ESWC'06. pp. 125–139. Springer (2006)
20. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: KR'10. AAI Press (2010)
21. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the Horn fragments of the description logics SHOIQ and SROIQ. In: IJCAI'11. pp. 1039–1044. IJCAI/AAAI (2011)
22. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. *J. Applied Logic* 8(2), 186–209 (2010)
23. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-Lite. In: DL'09. CEUR-WS.org (2009)
24. Rosati, R.: On conjunctive query answering in EL. In: DL'07. CEUR-WS.org (2007)
25. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR'10. AAI (2010)

Exact Query Reformulation over *SHOQ* DBoxes

Enrico Franconi, Volha Kerhet, Nhung Ngo

Free University of Bozen-Bolzano, Italy
lastname@inf.unibz.it

Abstract We formalise the problem of query reformulation over a description logic ontology and a DBox in a general framework. This framework supports deciding the existence of a safe-range first-order equivalent reformulation of a concept query in terms of the signature of a DBox. A constructive method to compute the reformulation is provided. We are particularly interested in safe-range reformulations since they can be transformed to relational queries and executed using SQL. We also discuss the completeness of the proposed framework with respect to finite and unrestricted models. As a case study we consider ontologies and queries expressed in *SHOQ*.

1 Introduction

In this paper we study and develop a query rewriting framework which is applicable to description logics systems where data is stored in a classical finite relational database, in a way that in the literature has been called *DBox* [5,6]. A DBox is a set of ground atoms which semantically behaves like a database, i.e., the interpretation of the database predicates in the DBox is exactly equal to the database relations. The DBox predicates are *closed*, i.e., their extensions are the same in every interpretation, whereas the other predicates in the knowledge base are *open*, i.e., their extensions may vary among different interpretations. We do not consider here the *open* interpretation for the database predicates – i.e., the classical *ABox*. In an *ABox*, the interpretation of database predicates contains the database relations and possibly more. This notion clearly is less faithful in the representation of a database semantics since it would allow for spurious interpretations of database predicates with additional unwanted tuples not present in the original database.

In our general framework an ontology is a TBox in a first-order description logic, and queries are concept expressions. Within this setting, the framework provides support to decide the existence of a relational algebra (i.e., safe-range first-order) *equivalent* (a.k.a. *exact*) reformulation of the query in terms of the DBox signature. It also provides an effective approach to construct the reformulation. We are particularly interested in safe-range reformulations of queries because their range-restricted syntax is needed to reduce the original query answering problem to a relational algebra evaluation (e.g., via SQL) over the original database [7]. Our framework points out several conditions on the ontology and the query in order to guarantee the existence of a safe-range equivalent reformulation. We show that these conditions are not infeasible in practice and we also provide an efficient method to ensure their validation. Standard tableau techniques can be used to compute the reformulation.

In order to be complete, our framework is applicable to ontologies and queries expressed in any fragment of first-order logic enjoying the finitely controllable determinacy [3,8]. If the employed logic does not enjoy the finitely controllable determinacy our approach would become sound but incomplete, by still effectively implementable using standard theorem proving techniques. We have explored non-trivial applications where the framework is complete; in this paper, the application with *SHOQ* ontology and concept queries is discussed. We show how (i) to check whether the answers to a given query with an ontology are *solely* determined by the extension of the DBox predicates and, if so, (ii) to find an equivalent rewriting of the query in terms of the DBox predicates to allow the use of standard database technology for answering the query. This means we benefit from the low computational complexity in the size of the data for answering queries on relational databases. In addition, it is possible to reuse standard techniques of description logics reasoning to find rewritings, such as in [5].

The query reformulation problem has received strong interest in classical relational database research as well as modern knowledge representation studies. Differently from the mainstream research on query reformulation [9], which is mostly based on perfect or maximally contained rewritings with sound views (see, e.g., the DL-Lite approach [10]), we focus here on exact rewritings with exact views, since it characterises more precisely the query answering problem with ontologies and databases, and it allows for very expressive ontology languages. An exact reformulation has the same answer in any model of the ontology with the DBox, and it provides a fully determined answer, which may be useful, e.g., for materialisation.

This work extends the seminal works on exact rewritings with exact views [2,5,3] by focussing on safe-range reformulations and on the conditions ensuring their existence in description logics. This is necessary when the description logic at hand is not enjoying the Beth definability property [11], which would guarantee the rewriting to be safe-range. The detailed algorithms and all the proofs for a more general framework are available in the technical report [8].

The paper is organised as follows. Section 2 provides the necessary formal background and definitions. Section 3 introduces a characterisation of the query reformulation problem, and the conditions allowing for an effective reformulation are analysed. At the end, we discuss in details the application to *SHOQ* ontologies with a DBox.

2 Preliminaries

In this section we define the basic concepts that are used in the paper.

2.1 Description Logics and DBox

Let N_C , N_R and N_I be sets of concept, role and individual names respectively. And let $\mathcal{L}(N_C, N_R, N_I)$ be some description logic language over N_C , N_R and N_I . An *ontology* is a set of TBox assertions in $\mathcal{L}(N_C, N_R, N_I)$.

Let C be a (possibly complex) concept or an assertion in $\mathcal{L}(N_C, N_R, N_I)$. We denote as $\sigma(C)$ the signature of C , that is the union of all concept, role and individual names occurring in C .

A DBox \mathcal{D} is a *finite* set of atomic concept and role assertions of the form $A(a)$ and $R(a, b)$ respectively, where $A \in N_C$, $R \in N_R$ and $a, b \in N_I$. The sets of all concept,

role and individual names appearing in \mathcal{D} are denoted as $\sigma_{\mathcal{D}}(C)$, $\sigma_{\mathcal{D}}(R)$ and $\sigma_{\mathcal{D}}(I)$ respectively. We call *DBox predicates* the set $\sigma_{\mathcal{D}}(P) = \sigma_{\mathcal{D}}(C) \cup \sigma_{\mathcal{D}}(R)$.

As usual, an *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ includes a domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps concepts to subsets of $\Delta^{\mathcal{I}}$, roles to binary relations on $\Delta^{\mathcal{I}}$ and individuals to elements of $\Delta^{\mathcal{I}}$.

We say that an interpretation \mathcal{I} *embeds a DBox* \mathcal{D} , written $\mathcal{I}_{(\mathcal{D})}$, if it holds that: (i) $a^{\mathcal{I}} = a$ for every DBox individual $a \in \sigma_{\mathcal{D}}(I)$, i.e. a follows the *standard name assumption* (SNA) [7]; (ii) for every concept name A in $\sigma_{\mathcal{D}}(C)$ and every $u \in \Delta^{\mathcal{I}}$, $u \in A^{\mathcal{I}}$ if and only if $A(u) \in \mathcal{D}$; and (iii) for every role name R in $\sigma_{\mathcal{D}}(R)$ and every pair $(u, v) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, $(u, v) \in R^{\mathcal{I}}$ if and only if $R(u, v) \in \mathcal{D}$. In other words, in every interpretation embedding \mathcal{D} , the interpretation of any DBox predicate is always the same and it is given exactly by its content in the DBox; this is, in general, not the case for the interpretation of the non-DBox predicates. Under above embedding conditions, we say that all the DBox predicates are *closed*, while all the other predicates are *open* and may be interpreted differently in different interpretations.

In order to allow for an arbitrary DBox to be embedded, we generalise the standard name assumption to all the individual names in N_I ; this implies that the domain of any interpretation necessarily includes the set of all the individual names N_I .

We denote an interpretation \mathcal{I} with a specific domain Δ as $\mathcal{I}^{(\Delta)}$. Given an interpretation \mathcal{I} , we denote as $\mathcal{I}|_{\mathbb{S}}$ the interpretation restricted to the smaller signature $\mathbb{S} \subseteq N_C \cup N_R \cup N_I$, i.e., the interpretation with the same domain $\Delta^{\mathcal{I}}$ and the same interpretation function $\cdot^{\mathcal{I}}$ defined only for the concept, role and individual names from the set \mathbb{S} .

We call $FOL(\mathbb{C}, \mathbb{P})$ a function free first-order language with equality over a signature $\Sigma = (\mathbb{C}, \mathbb{P})$, where $\mathbb{C} = N_I$ is a set of constants and $\mathbb{P} = N_C \cup N_R$ is a set of predicates with arities 1 (for concept names) and 2 (for role names).

An interpretation in which an assertion φ (TBox or ABox) is true is called a *model* of φ ; the set of all models of φ is denoted as $M(\varphi)$. The set of all models of all assertions in an ontology \mathcal{T} is denoted as $M(\mathcal{T})$. We say that a *DBox* \mathcal{D} is *legal for an ontology* \mathcal{T} if there exists a model of \mathcal{T} embedding \mathcal{D} . In the paper, we consider only consistent non-tautological ontologies and legal DBoxes.

2.2 Queries and certain answers

A *query* is a concept in $\mathcal{L}(N_C, N_R, N_I)$. Given a query Q , we define its *certain answer* to a DBox \mathcal{D} under \mathcal{T} as follows:

Definition 1 (Certain Answer) *The (certain) answer of a query Q to a DBox \mathcal{D} under an ontology \mathcal{T} is the set of individuals:*

$$\{a \in N_I \mid \forall \mathcal{I}_{(\mathcal{D})} \in M(\mathcal{T}) : \mathcal{I}_{(\mathcal{D})} \models Q(a)\}.$$

We now show that we can weaken the standard name assumption for the constants by just assuming *unique names*, without changing the certain answers. As we said before, an interpretation \mathcal{I} embedding a DBox \mathcal{D} satisfies the standard name assumption – written $\mathcal{I}_{(\mathcal{D})}^{SNA}$ – if $c^{\mathcal{I}} = c$ for any $c \in N_I$. Alternatively, an interpretation \mathcal{I} embedding a DBox \mathcal{D} satisfies the unique name assumption – written $\mathcal{I}_{(\mathcal{D})}^{UNA}$ – if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for any different $a, b \in N_I$. The following proposition allows us to freely interchange the

standard name and the unique name assumptions in dealing with interpretations embedding DBoxes. This is a practical advantage, since most description logics reasoners do have a native unique name assumption.

Proposition 1 (SNA vs UNA) For any query $Q(x)$, ontology \mathcal{T} and DBox \mathcal{D} ,

$$\{a \in N_I \mid \forall \mathcal{I}_{(\mathcal{D})}^{\text{SNA}} \in M(\mathcal{T}) : \mathcal{I}_{(\mathcal{D})}^{\text{SNA}} \models Q(a)\} = \{a \in N_I \mid \forall \mathcal{I}_{(\mathcal{D})}^{\text{UNA}} \in M(\mathcal{T}) : \mathcal{I}_{(\mathcal{D})}^{\text{UNA}} \models Q(a)\}.$$

A query is *DBox-relativised* if and only if its answer is bounded by the DBox.

Definition 2 (DBox-relativised query) A concept query Q is DBox-relativised under ontology \mathcal{T} , if in each model of \mathcal{T} the interpretation of Q includes only domain elements which are among the interpretation of DBox predicates or of individuals from \mathcal{T} or Q .

2.3 Safe-range formulas

Since a query can be an arbitrary first-order formula, its answer can be infinite (since the domain is not restricted to be finite) or it may depend on the domain. To eliminate such cases, we will consider *domain independent* queries. For example, the query $Q = \neg \text{Student}$ over the DBox $\text{Student}(A), \text{Student}(B)$, with domain $\{A, B, C\}$ has the answer $\{x = C\}$, with domain $\{A, B, C, D\}$ has the answer $\{x = C, x = D\}$, and if we change the domain to an infinite one, the answer will be infinite even in presence of such a finite database. Therefore, the notion of *domain independent* queries has been introduced in relational databases.

In general, the problem of checking whether a FOL formula is domain independent is undecidable [7]. The well known *safe-range* syntactic fragment of FOL introduced by Codd is an *equally expressive* language; indeed any safe-range formula is domain independent, and any domain independent formula can be easily transformed into a logically equivalent safe-range formula. Intuitively, a formula is safe-range if and only if its variables are bounded by positive predicates or equalities – for the exact syntactical definition see, e.g., [7]. For example, the formula $\neg A(x) \wedge B(x)$ is safe-range, while queries $\neg A(x)$ and $\forall x. A(x)$ are not. To check whether a formula is safe-range, the formula is transformed into a logically equivalent *safe-range normal form* and its *range restriction* is computed according to a set of syntax based rules; the range restriction of a formula is a subset of its free variables, and if it coincides with the free variables then the formula is said to be safe-range.

Any formula in $FOL(\mathbb{C}, \mathbb{P})$ can be transformed to a logically equivalent *safe range normal form* (SRNF) by recursively applying the following steps :

- Variable substitution: no distinct pair of quantifiers may employ same variable.
- Elimination of universal quantifiers
- Elimination of implications
- Pushing negation
- Flattening of and/or

A formula is said to be SRNF if none of the aforementioned steps can be applied any more. Let φ be a formula in $FOL(\mathbb{C}, \mathbb{P})$, we denote the set of all variables appearing in φ as $\text{VAR}(\varphi)$, and the set of the free variables appearing in φ as $\text{FREE}(\varphi)$. The safe

range normal form of φ is denoted as $\text{SRNF}(\varphi)$. Let φ be a formula in SRNF. The *range restriction* of φ , denoted as $rr(\varphi)$, is either a subset of $\text{FREE}(\varphi)$ or \perp , and it is computed according to the following rules:

- $rr(R(t_1, \dots, t_n)) = \text{VAR}(R(t_1, \dots, t_n))$;
- $rr(x = y) = \emptyset$;
- $rr(x = c) = \{x\}$, where $c \in \mathbb{C}$;
- $rr(\varphi_1 \wedge \varphi_2) = rr(\varphi_1) \cup rr(\varphi_2)$;
- $rr(\varphi_1 \vee \varphi_2) = rr(\varphi_1) \cap rr(\varphi_2)$;
- $rr(\varphi \wedge x = y) = rr(\varphi)$, if $\{x, y\} \cap rr(\varphi) = \emptyset$; and $rr(\varphi \wedge x = y) = rr(\varphi) \cup \{x, y\}$ otherwise;
- $rr(\neg\varphi) = \emptyset \cap rr(\varphi)$;
- $rr(\exists x\varphi) = rr(\varphi) \setminus x$ if $x \in rr(\varphi)$ and $rr(\exists x\varphi) = \perp$ otherwise.

We consider \perp as a special set, such that for any set Z : $\perp \cup Z = \perp \cap Z = \perp \setminus Z = Z \setminus \perp = \perp$. If φ is not in SRNF, then $rr(\varphi) := rr(\text{SRNF}(\varphi))$. We say that a *variable* $x \in \text{FREE}(\varphi)$ has *restricted range* in φ if $x \in rr(\varphi)$.

Definition 3 (Safe range) A formula φ is *safe range* iff $rr(\text{SRNF}(\varphi)) = \text{FREE}(\varphi)$.

We also consider a weaker version of safe-range property called ground safe-range. Given a formula, its *grounding* is the formula itself where all free variables are replaced by new constants.

Definition 4 (Ground safe-range) A formula is *ground safe-range* if its *grounding* is *safe-range*.

The safe-range fragment of first-order logic with the standard name assumption is equally expressive to the relational algebra, which is the core of the SQL query language [7].

For any concept C in $\mathcal{L}(N_C, N_R, N_I)$ we denote the corresponding logically equivalent formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ with one free variable x as $C(x)$. We will call any axiom (concept) in $\mathcal{L}(N_C, N_R, N_I)$ (ground) safe-range, if the corresponding logically equivalent formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ is (ground) safe-range. An ontology \mathcal{T} in $\mathcal{L}(N_C, N_R, N_I)$ is safe-range, if every formula in \mathcal{T} is safe-range.

3 Exact Safe-range Query Reformulation

In this section we state the problem of finding a first-order safe-range reformulation of a concept query. We then find the conditions to reduce the original query answering problem – which corresponds to an entailment problem – to a model checking problem of the reformulation over the DBox.

Let us consider the class of queries of interest. The certain answer to a query includes all the individuals which make the query true in *all* the models of the ontology: so, if an individual would make the query true only in some model, then it would be discarded from the certain answer. In other words, it may be the case that the answer to the query is not necessarily the same among all the models of the knowledge base. In this case, the query is not fully determined by the given source data; indeed, there

is some answer which is possible, but not certain. Due to the indeterminacy of the data wrt the query, the complexity to compute the certain answer in general increases, and it corresponds to the complexity of entailment in the logic. In this paper we focus on the case when a query has the same answer over all the models of the ontology, namely, on the case when the information requested by the query is fully available from the source data without ambiguity. In this way, the indeterminacy disappears, and the complexity of the process may decrease.

A query is *definable* [12] if its truth value in any model of the ontology depends *only* on the domain and on the interpretation of the database predicates and constants. The answer of a definable query does not depend on the interpretation of non-database predicates. Once the database and a domain are fixed, it is never the case that an individual would make the query true in some model of the knowledge base and false in others, since the truth value of an implicitly defined query depends only on the interpretation of the database predicates and constants and on the domain (which are fixed).

[12] proved that, in first-order logic, looking for definable queries from the DBox predicates amounts at having an *exact reformulation* of the query in terms of the DBox predicates.

Definition 5 (Exact Reformulation) *The $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ formula \widehat{Q} is an exact reformulation of Q under \mathcal{T} over $\sigma_{\mathcal{D}}(P)$ if $\sigma(\widehat{Q}) \subseteq \sigma_{\mathcal{D}}(P)$ and $\mathcal{T} \models \forall x.Q(x) \leftrightarrow \widehat{Q}(x)$.*

Since we are dealing with finite databases, in the following we will focus on those fragments of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ for which the exact reformulation over unrestricted models and over finite models coincide; we say that these fragments have *finitely controllable determinacy*.

Given DBox predicates $\sigma_{\mathcal{D}}(P)$, an ontology \mathcal{T} , and a query Q in the $\mathcal{L}(N_C, N_R, N_I)$ language, our goal is to find a safe-range exact reformulation \widehat{Q} of Q in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ expressed in terms of DBox predicates, that being evaluated as a relational algebra expression over a legal DBox (e.g., using a relational database system with SQL) gives the same answer as the certain answer of Q to the DBox under \mathcal{T} .

Since an exact reformulation is equivalent under the ontology to the original query, the certain answer of the original query and of the reformulated query are identical. More precisely, the following proposition holds.

Proposition 2 *Let \widehat{Q} be an exact reformulation of Q under \mathcal{T} over $\sigma_{\mathcal{D}}(P)$, then:*

$$\begin{aligned} & \{a \in N_I \mid \forall \mathcal{I}_{(\mathcal{D})} \in M(\mathcal{T}) : \mathcal{I}_{(\mathcal{D})} \models Q(a)\} = \\ & \{a \in N_I \mid \forall \mathcal{I}_{(\mathcal{D})} \in M(\mathcal{T}) : \mathcal{I}_{(\mathcal{D})} \models \widehat{Q}(a)\}. \end{aligned}$$

From the above equation it is clear that in order to answer an exactly reformulated query, one still may need to consider all the models $\mathcal{I}_{(\mathcal{D})}$ of the ontology embedding the DBox – i.e., we still have an entailment problem to solve. The following theorem states the condition to reduce the original query answering problem – based on entailment – to the problem of checking the validity of the exact reformulation over a *single* model: the condition is that the reformulation should be safe-range.

Theorem 1 (Adequacy of Exact safe-range Query Reformulation) *Let \mathcal{T} be an ontology in $\mathcal{L}(N_C, N_R, N_I)$, Q be a query in $\mathcal{L}(N_C, N_R, N_I)$ and \mathcal{D} be a legal DBox for \mathcal{T} . If \widehat{Q} is an exact reformulation of Q under \mathcal{T} over $\sigma_{\mathcal{D}}(P)$ and \widehat{Q} is safe-range, then:*

$$\begin{aligned} & \{a \in N_I \mid \forall \mathcal{I}_{(\mathcal{D})} \in M(\mathcal{T}) : \mathcal{I}_{(\mathcal{D})} \models Q(a)\} = \\ & \{a \in \text{adom}(\sigma(\widehat{Q}), \mathcal{D}) \mid \mathcal{I}_{(\mathcal{D})}^{(N_I)} \upharpoonright_{\sigma_{\mathcal{D}}(P) \cup N_I} \models \widehat{Q}(a)\}, \end{aligned}$$

where $\text{adom}(\sigma(\widehat{Q}), \mathcal{D})$ consists of all the constants from \widehat{Q} and from the assertions in \mathcal{D} corresponding to concept and role names appearing in \widehat{Q} .

A safe-range reformulation is *necessary* to transform a first-order query to a relational algebra query which can then be evaluated by using SQL techniques. The theorem above shows in addition that being safe-range is also a *sufficient* property for an exact reformulation to be correctly evaluated as an SQL query.

However, given an arbitrary input (an ontology, a DBox and a concept query), one can not guarantee the existence of an exact safe-range reformulation. Therefore, in the rest of this section we introduce conditions on the input to get an exact safe-range reformulation. Moreover, since we are dealing with a finite DBox, we have also to consider the condition under which the existence of an exact safe-range reformulation under unrestricted reasoning and under finite reasoning coincide.

Let Q be any formula and \widetilde{Q} the formula obtained from it by uniformly replacing every occurrence of each non-DBox predicate P with a new predicate \widetilde{P} . We extend this renaming operator $\widetilde{\cdot}$ to any set of formulas in a natural way. Then the following *constructive theorem* gives us sufficient conditions to check the existence of an exact safe-range reformulation.

Theorem 2 (Constructive Theorem) *If the following conditions hold:*

1. $\mathcal{T} \cup \widetilde{\mathcal{T}} \models Q \equiv \widetilde{Q}$;
2. Q is DBox-relativised under \mathcal{T} ;
3. Q is ground safe-range;
4. \mathcal{T} is safe-range;

then there exists an exact safe-range reformulation \widehat{Q} of Q in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ over $\sigma_{\mathcal{D}}(P)$ under \mathcal{T} .

The above conditions can be divided into two groups: the first condition forces the existence of an exact reformulation, while the three last conditions guarantee its safe-range property. The first condition says that one does not need to consider non-DBox predicates to answer the query. In other words, its answer in any model of the ontology depends *only* on the domain and on the interpretation of the DBox predicates and constants. This property of a query is called *implicit definability* from a set of predicates (the DBox predicates) in first-order logic [12]. The second condition points out that the answer of the query is necessarily in the set of individuals appearing in the DBox original query or ontology.

The first two conditions are necessary to have an exact safe-range reformulation, i.e. if there is an exact safe-range reformulation, then the original query should be implicitly definable and DBox-relativised. The last two conditions are just sufficient ones, as the following example shows.

Example 1 *Let $\mathbb{P} = \{A, B, C\}$, $\sigma_{\mathcal{D}}(I) = \{C\}$, $\mathcal{T} = \{A \equiv C, \top \sqsubseteq B\}$, $Q = A \sqcap B$.*

- Q is implicitly definable from the DBox predicates under \mathcal{T} because the first assertion of \mathcal{T} gives an explicit definition of Q ;

- Q is safe-range;
- Q is DBox-relativised under \mathcal{T} because of the first assertion of \mathcal{T} .
- $\tilde{Q}(x) = C$ is an exact safe-range reformulation of Q under \mathcal{T} over $\sigma_{\mathcal{D}}(I)$.

But \mathcal{T} is not safe-range because of the second assertion. □

4 A case study: \mathcal{SHOQ}

Syntax	Semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\exists R.C$	$\{x \mid \text{exists } y \text{ such that } (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
$\forall R.C$	$\{x \mid \text{forall } y (x, y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$
$\{o\}$	$\{o\}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\geq nR.C$	$\{x \mid \#(\{y \mid (x, y) \in R^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \geq n\}$
$\leq nR.C$	$\{x \mid \#(\{y \mid (x, y) \in R^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \leq n\}$

Table 1. Syntax and semantics of \mathcal{SHOQ} concepts

\mathcal{SHOQ} is an extension of the description logic \mathcal{ALC} with transitive roles, role hierarchies, qualified number restrictions, and individuals; it is a fragment of first-order logic and of OWL2. The syntax and semantics of \mathcal{SHOQ} is summarised in table 1, where A is an atomic concept, C and D are concepts, o is an individual name and R is an atomic role. \mathcal{SHOQ} is a pretty much standard description logic; for more details see, e.g., [13]. A TBox in \mathcal{SHOQ} is a set of concept inclusion axioms $C \sqsubseteq D$, role inclusion axioms $R \sqsubseteq S$, and transitivity axioms $\text{Trans}(R)$ (where C, D are concepts and R, S are atomic roles) with the usual expected semantics.

In this section, we present an application of our framework where the ontology is a TBox in \mathcal{SHOQ} , and the query is a \mathcal{SHOQ} concept.

Finitely controllable determinacy. Does \mathcal{SHOQ} have finite controllability of determinacy? It is enough to check that the entailment $\mathcal{T} \cup \tilde{\mathcal{T}} \models Q \equiv \tilde{Q}$ coincide in the unrestricted and finite cases. The finite controllability of this equivalence axiom entailment in \mathcal{SHOQ} is guaranteed because of the two following reasons:

- The entailment $\mathcal{T} \cup \tilde{\mathcal{T}} \models Q \equiv \tilde{Q}$ can be reduced in \mathcal{SHOQ} to a concept satisfiability problem for an empty TBox.
- \mathcal{SHOQ} has finite model property [14].

So, we can use a standard \mathcal{SHOQ} reasoner (e.g., an OWL2 reasoner) to check the first condition.

Safe-range ontology. Let's now check whether a \mathcal{SHOQ} ontology is safe-range. Role inclusion and transitivity axioms are always safe-range. Unfortunately, concept inclusion axioms in \mathcal{SHOQ} ontologies may not be safe-range: for example, the axiom $\neg \text{male} \sqsubseteq \text{female}$ is not safe-range. It is easy to see that an axiom $C \sqsubseteq D$ is not safe-range if and only if $C(x)$ is not safe-range and $D(x)$ is safe-range: just observe that the axiom is logically equivalent to the formula $\neg \exists x. C(x) \wedge \neg D(x)$ in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$. The following proposition provides rules deciding whether a \mathcal{SHOQ} concept is safe-range.

Proposition 3 *Let A be an atomic concept, C and D be \mathcal{SHOQ} concepts. Then the open formulas:*

1. $A(x), (\exists R.C)(x), \{o\}(x), (\geq nR.C)(x)$ are safe-range;
2. $(\forall R.C)(x), (\leq nR.C)(x)$ are not safe-range;
3. $(C \sqcap D)(x)$ is safe-range if and only if $C(x)$ is safe-range or $D(x)$ is safe-range;
4. $(C \sqcup D)(x)$ is safe-range if and only if $C(x)$ is safe-range and $D(x)$ is safe-range;
5. $\neg C(x)$ is safe-range if and only if $C(x)$ is not safe-range.

Proposition 4 *For any \mathcal{SHOQ} concept C , $C(x)$ is ground safe-range.*

The presence of non-safe-range axioms in an ontology would prevent the application of our framework, but we argue that non-safe-range axioms should not appear in a cleanly designed \mathcal{SHOQ} ontology, and, if present, they should be fixed. Indeed, the use of *absolute* negative information in the subsumee – such as in the axiom “a non-male is a female” ($\neg \text{male} \sqsubseteq \text{female}$) – should be deprecated by a clean design methodology, since the subsumer would include *all sorts* of objects in the universe (but the ones of the subsumee type) without any obvious control. Only *relativised* negative information in the subsumee should be allowed – such as in the axiom “a non-male person is a female” ($\text{person} \sqcap \neg \text{male} \sqsubseteq \text{female}$). This observations suggests a fix for non-safe-range axioms: for every non-safe-range axiom $C \sqsubseteq D$ users will be asked to replace it by the safe-range one $C \sqcap E \sqsubseteq D$, where E is an arbitrary safe-range concept. Therefore, the user is asked to make explicit the *type* of the subsumee, in a way to make it safe-range; note that the type could be also a fresh new atomic concept. We believe that the fix we are proposing for \mathcal{SHOQ} is a reasonable one, and would make all \mathcal{SHOQ} ontologies eligible to be used with our framework.

Ground safe-range and DBox-relativised query. Let \mathcal{T} be a \mathcal{SHOQ} ontology, and Q an implicitly definable query, which is a possibly complex concept in \mathcal{SHOQ} . In order to be able to use our framework, a query should be ground safe-range and *DBox*-relativised under the ontology. We already know by proposition 4 that a concept query is always ground safe-range. A query is *DBox*-relativised if it returns only *DBox* individuals; it may be strange for a user to issue a query which is not meant to return just *DBox* objects. One can check if Q is *DBox*-relativised under the ontology by using the following proposition.

Proposition 5 *The query Q is *DBox*-relativised under \mathcal{T} if and only if:*

$$\mathcal{T} \models_{\mathcal{SHOQ}} Q \sqsubseteq \bigsqcup_{i=1}^k \{o_i\} \sqcup \bigsqcup_{i=1}^n A_i \sqcup \bigsqcup_{i=1}^m (\exists R_i. \top \sqcup \exists R_i^-. \top), \quad (1)$$

where $\{A_1, \dots, A_n\}$ is the set of all DBox concepts appearing in \mathcal{T} and \mathcal{Q} ;
 $\{R_1, \dots, R_m\}$ is the set of all DBox roles appearing in \mathcal{T} and \mathcal{Q} ; and
 $\{o_1, \dots, o_k\}$ is the set of all individual names appearing in \mathcal{T} and \mathcal{Q} .

In other words, if the *SHOIQ* entailment in the proposition is valid, then the query is *DBox-relativised* under the ontology. We use *SHOIQ* instead of *SHOQ* because we need inverse roles. Due to the incompleteness wrt finite model reasoning of the *SHOIQ* test, one might conclude that a query is not *DBox-relativised* but in fact it is *DBox-relativised* under finite model reasoning. In the rare case a user is issuing a real non-*DBox-relativised* query, or a *DBox-relativised* query which failed the above test due to its incompleteness, we would ask the user to conjoin the query with a safe-range concept composed only by database atomic concepts, which would become the *type* of the query. We believe that also this fix for the queries is a reasonable one, and would make all queries eligible to be used with our framework.

A complete procedure. Given a *SHOQ* ontology \mathcal{T} , a legal DBox \mathcal{D} and a concept query \mathcal{Q} , one can apply the procedure below to generate a safe-range exact reformulation over the DBox predicates.

Input: A *SHOQ* TBox \mathcal{T} , a concept query Q in *SHOQ* and a DBox predicates (DBox atomic concepts and roles).

1. Check implicit definability of the query Q by testing $\mathcal{T} \cup \tilde{\mathcal{T}} \models Q \equiv \tilde{Q}$ using standard DL reasoner of *SHOQ*. If it is the case, continue.
2. Check whether \mathcal{T} is safe-range, and fix it if it is not safe-range.
3. Check the DBox-relativisation of Q , and fix it if it is not DBox-relativised.
4. Use the constructive theorem to
 - (a) compute a ground safe-range reformulation $Q'(x)$ from the tableau proof generated in step 1 (this is an extension of what has been presented in [5,11]; see [8] for a complete characterisation);
 - (b) transform it to a safe-range one as follows: $\hat{Q}(x) := Q'(x) \wedge ADOM(x)$, where $ADOM$ is a predicate containing all the individuals in the DBox, in \mathcal{T} , and in \mathcal{Q} . $ADOM$ actually represents the subsumer of the TBox axiom in (1).

Output: A safe-range first-order exact reformulation $\hat{Q}(x)$ expressed over the DBox predicates.

Note that the above procedure could be executed once for all at compile time: indeed, it could be run for each atomic concept in the ontology, and the outcome for each of them could be stored persistently, if the reformulation has been successful.

5 Conclusion

We have introduced a framework to compute the exact reformulation of concept queries to a DBox in description logics. We have found the conditions which guarantee that a safe-range reformulation exists, and we show that it can be evaluated as a relational algebra query over the database to give the same answer as the original query under the ontology. A non-trivial case study has been presented in the field of description logics, with the *SHOQ* language.

As a future work, we would like to study optimisations of reformulations. From the practical perspective, since there might be many rewritten queries from one original query, the problem of selecting an optimised one in terms of query evaluation is very important. In fact, one has to take into account which criteria should be used to optimise, such as: the size of the rewritings, the numbers of used predicates, the priority of predicates, the number of relational operators, and clever usage of duplicates.

We wish to thank David Toman, İnanç Seylan, Jos de Bruijn, Alex Borgida, Grant Weddell, Tommaso Di Noia, Umberto Straccia, Balder ten Cate with whom we have learnt a lot about query rewriting based on Beth definability, and anonymous reviewers for insightful comments.

References

1. Etzioni, O., Golden, K., Weld, D.S.: Sound and efficient closed-world reasoning for planning. *Artif. Intell.* **89** (January 1997) 113–148
2. Marx, M.: Queries determined by views: pack your views. In: *Proceedings of the 26th ACM symposium on Principles of Database Systems. PODS '07 (2007)* 23–30
3. Nash, A., Segoufin, L., Vianu, V.: Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.* **35** (July 2010) 21:1–21:41
4. Fan, W., Geerts, F., Zheng, L.: View determinacy for preserving selected information in data transformations. *Inf. Syst.* **37** (March 2012) 1–12
5. İnanç Seylan, Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*. (2009) 923–925
6. Franconi, E., Ibanez-Garcia, Y.A., İnanç Seylan: Query answering with DBoxes is hard. *Electronic Notes in Theoretical Computer Science, Elsevier* **278** (November 2011) 71–84
7. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*. Addison-Wesley (1995)
8. Franconi, E., Kerhet, V., Ngo, N.: Exact query reformulation with expressive ontologies and databases. Technical Report 12158, KRDB Tech research group, Free University of Bozen-Bolzano (March 2012) <http://www.inf.unibz.it/krdb/pub/TR/KRDB-Tech-12158.pdf>.
9. Halevy, A.Y.: Answering queries using views: A survey. *The VLDB Journal* **10** (December 2001) 270–294
10. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: The DL-Lite family and relations. *J. Artif. Intell. Res. (JAIR)* **36** (2009) 1–69
11. ten Cate, B., Franconi, E., İnanç Seylan: Beth definability in expressive description logics. In: *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*. (2011) 1099–1106
12. Beth, E.: On Padoa's method in the theory of definition. *Indagationes Mathematicae* **15** (1953) 330–339
13. Horrocks, I., Sattler, U.: Ontology reasoning in the SHOQ(D) description logic. In: *In Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*. (2001) 199–204
14. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. *J. Artif. Int. Res.* **23** (June 2005) 667–726

Preferential Low Complexity Description Logics: Complexity Results and Proof Methods

Laura Giordano¹, Valentina Gliozzi², Nicola Olivetti³, and Gian Luca Pozzato²

¹ Dip. di Informatica - U. Piemonte O. - Alessandria - Italy - laura@mf.n.unipmn.it

² Dip. Informatica - Univ. di Torino - Italy - {gliozzi,pozzato}@di.unito.it

³ LSIS-UMR CNRS 6168 - Marseille - France - nicola.olivetti@univ-cezanne.fr

Abstract. In this paper we describe an approach for reasoning about typicality and defeasible properties in low complexity preferential Description Logics. We describe the non-monotonic extension of the low complexity DLs \mathcal{EL}^\perp and $DL-Lite_{core}$ based on a typicality operator \mathbf{T} , which enjoys a preferential semantics. We summarize complexity results for such extensions, called $\mathcal{EL}^\perp \mathbf{T}_{min}$ and $DL-Lite_c \mathbf{T}_{min}$. Entailment in $DL-Lite_c \mathbf{T}_{min}$ is in Π_2^P , whereas entailment in $\mathcal{EL}^\perp \mathbf{T}_{min}$ is EXPTIME-hard. However, for the Left Local fragment of $\mathcal{EL}^\perp \mathbf{T}_{min}$ the complexity of entailment drops to Π_2^P . We present tableau calculi for Left Local $\mathcal{EL}^\perp \mathbf{T}_{min}$ and for $DL-Lite_c \mathbf{T}_{min}$. The calculi perform a two-phase computation in order to check whether a query is minimally entailed from the initial knowledge base. The calculi are sound, complete and terminating, and provide decision procedures for verifying entailment in the two logics, whose complexities match the above mentioned complexity results.

1 Introduction

Nonmonotonic extensions of Description Logics (DLs) have been actively investigated since the early 90s [15, 4, 2, 3, 7, 12, 8, 6]. A simple but powerful non-monotonic extension of DLs is proposed in [12, 8]: in this approach “typical” or “normal” properties can be directly specified by means of a “typicality” operator \mathbf{T} enriching the underlying DL; the typicality operator \mathbf{T} is essentially characterised by the core properties of non-monotonic reasoning axiomatized by *preferential logic* [13]. In $\mathcal{ALC} + \mathbf{T}$ [12], one can consistently express defeasible inclusions and exceptions such as: typical students do not pay taxes, but working students do typically pay taxes, but working students having children normally do not: $\mathbf{T}(Student) \sqsubseteq \neg TaxPayer$; $\mathbf{T}(Student \sqcap Worker) \sqsubseteq TaxPayer$; $\mathbf{T}(Student \sqcap Worker \sqcap \exists HasChild.\top) \sqsubseteq \neg TaxPayer$. Although the operator \mathbf{T} is non-monotonic in itself, the logic $\mathcal{ALC} + \mathbf{T}$ is monotonic. As a consequence, unless a KB contains explicit assumptions about typicality of individuals (e.g. that john is a typical student), there is no way of inferring defeasible properties of them (e.g. that john does not pay taxes). In [8], a non-monotonic extension of $\mathcal{ALC} + \mathbf{T}$ based on a minimal model semantics is proposed. The resulting logic, called $\mathcal{ALC} + \mathbf{T}_{min}$, supports typicality assumptions, so that if one knows that john is a student, one can non-monotonically assume that he is also a *typical* student and therefore that he does not pay taxes. As an example, for a TBox specified by the inclusions above, in $\mathcal{ALC} + \mathbf{T}_{min}$ the following inference holds: $TBox \cup \{Student(john)\} \models_{\mathcal{ALC} + \mathbf{T}_{min}} \neg TaxPayer(john)$.

Similarly to other non-monotonic DLs, adding the typicality operator with its minimal model semantics to a standard DL, such as \mathcal{ALC} , leads to a very high complexity (namely, query entailment in $\mathcal{ALC} + \mathbf{T}_{min}$ is in $CO-NEXP^{NP}$ [8]). This fact

has motivated the study of non-monotonic extensions of low complexity DLs such as $DL-Lite_{core}$ [5] and \mathcal{EL}^\perp of the \mathcal{EL} family [1] which are nonetheless well-suited for encoding large knowledge bases (KBs).

In this paper, we consider the extensions of the low complexity logics $DL-Lite_{core}$ and \mathcal{EL}^\perp with the typicality operator based on the minimal model semantics introduced in [8]. We summarize complexity upper bounds for the resulting logics $\mathcal{EL}^\perp \mathbf{T}_{min}$ and $DL-Lite_c \mathbf{T}_{min}$ given in [11]. For \mathcal{EL}^\perp , it turns out that its extension $\mathcal{EL}^\perp \mathbf{T}_{min}$ is unfortunately EXPTIME-hard. This result is analogous to the one for *circumscribed* \mathcal{EL}^\perp KBs [3]. However, the complexity decreases to Π_2^p for the fragment of *Left Local* \mathcal{EL}^\perp KBs, corresponding to the homonymous fragment in [3]. The same complexity upper bound is obtained for $DL-Lite_c \mathbf{T}_{min}$.

We also describe the tableau calculi for $DL-Lite_c \mathbf{T}_{min}$ as well as for the Left Local fragment of $\mathcal{EL}^\perp \mathbf{T}_{min}$ for deciding minimal entailment in Π_2^p . Our calculi perform a two-phase computation: in the first phase, candidate models (complete open branches) falsifying the given query are generated, in the second phase the minimality of candidate models is checked by means of an auxiliary tableau construction. The calculi do not require any blocking machinery in order to achieve termination. A reformulation of existential rules, together with the idea of constructing multilinear models, is sufficient to match the Π_2^p complexity.

2 The Typicality Operator \mathbf{T} and the Logic $\mathcal{EL}^\perp \mathbf{T}_{min}$

Before describing $\mathcal{EL}^\perp \mathbf{T}_{min}$, let us briefly recall the underlying monotonic logic $\mathcal{EL}^{++} \mathbf{T}$, obtained by adding to \mathcal{EL}^\perp the typicality operator \mathbf{T} . The intuitive idea is that $\mathbf{T}(C)$ selects the *typical* instances of a concept C . In $\mathcal{EL}^{++} \mathbf{T}$ we can therefore distinguish between the properties that hold for all instances of concept C ($C \sqsubseteq D$), and those that only hold for the normal or typical instances of C ($\mathbf{T}(C) \sqsubseteq D$).

Formally, the $\mathcal{EL}^{++} \mathbf{T}$ language is defined as follows.

Definition 1. We consider an alphabet of concept names \mathcal{C} , of role names \mathcal{R} , and of individuals \mathcal{O} . Given $A \in \mathcal{C}$ and $R \in \mathcal{R}$, we define

$$C := A \mid \top \mid \perp \mid C \sqcap C \quad C_R := C \mid C_R \sqcap C_R \mid \exists R.C \quad C_L := C_R \mid \mathbf{T}(C)$$

A KB is a pair (TBox, ABox). TBox contains a finite set of general concept inclusions (or subsumptions) $C_L \sqsubseteq C_R$. ABox contains assertions of the form $C_L(a)$ and $R(a, b)$, where $a, b \in \mathcal{O}$.

The semantics of $\mathcal{EL}^{++} \mathbf{T}$ is defined by enriching ordinary models of \mathcal{EL}^\perp by a *preference relation* $<$ on the domain, whose intuitive meaning is to compare the “typicality” of individuals: $x < y$, means that x is more typical than y . Typical members of a concept C , that is members of $\mathbf{T}(C)$, are the members x of C that are minimal with respect to this preference relation.

Definition 2 (Semantics of \mathbf{T}). A model \mathcal{M} is any structure $\langle \Delta, <, I \rangle$ where Δ is the domain; $<$ is an irreflexive and transitive relation over Δ that satisfies the following Smoothness Condition: for all $S \subseteq \Delta$, for all $x \in S$, either $x \in \text{Min}_{<}(S)$ or $\exists y \in \text{Min}_{<}(S)$ such that $y < x$, where $\text{Min}_{<}(S) = \{u : u \in S \text{ and } \nexists z \in S \text{ s.t. } z < u\}$.

Furthermore, $<$ is multilinear: if $u < z$ and $v < z$, then either $u = v$ or $u < v$ or $v < u$. I is the extension function that maps each concept C to $C^I \subseteq \Delta$, and each role r to $r^I \subseteq \Delta^I \times \Delta^I$. For concepts of \mathcal{EL}^\perp , C^I is defined in the usual way. For the \mathbf{T} operator: $(\mathbf{T}(C))^I = \text{Min}_{<}(C^I)$.

Given a model \mathcal{M} , I can be extended so that it assigns to each individual a of \mathcal{O} a distinct element a^I of the domain Δ . We say that \mathcal{M} satisfies an inclusion $C \sqsubseteq D$ if $C^I \subseteq D^I$, and that \mathcal{M} satisfies $C(a)$ if $a^I \in C^I$ and aRb if $(a^I, b^I) \in R^I$. Moreover, \mathcal{M} satisfies TBox if it satisfies all its inclusions, and \mathcal{M} satisfies ABox if it satisfies all its formulas. \mathcal{M} satisfies a KB (TBox, ABox), if it satisfies both its TBox and its ABox.

The operator \mathbf{T} [12] is characterized by a set of postulates that are essentially a reformulation of the KLM [13] axioms of *preferential logic* \mathbf{P} . \mathbf{T} has therefore all the “core” properties of non-monotonic reasoning as it is axiomatized by \mathbf{P} . The semantics of the typicality operator can be specified by modal logic. The interpretation of \mathbf{T} can be split into two parts: for any x of the domain Δ , $x \in (\mathbf{T}(C))^I$ just in case (i) $x \in C^I$, and (ii) there is no $y \in C^I$ such that $y < x$. Condition (ii) can be represented by means of an additional modality \square , whose semantics is given by the preference relation $<$ interpreted as an accessibility relation. The interpretation of \square in \mathcal{M} is as follows: $(\square C)^I = \{x \in \Delta \mid \text{for every } y \in \Delta, \text{ if } y < x \text{ then } y \in C^I\}$. We immediately get that $x \in (\mathbf{T}(C))^I$ if and only if $x \in (C \square \square \neg C)^I$. From now on, we consider $\mathbf{T}(C)$ as an abbreviation for $C \square \square \neg C$.

As mentioned in the Introduction, the main limit of $\mathcal{EL}^{\perp+} \mathbf{T}$ is that it is *monotonic*. Even if the typicality operator \mathbf{T} itself is non-monotonic (i.e. $\mathbf{T}(C) \sqsubseteq E$ does not imply $\mathbf{T}(C \square D) \sqsubseteq E$), what is inferred from an $\mathcal{EL}^{\perp+} \mathbf{T}$ KB can still be inferred from any KB' with $\text{KB} \subseteq \text{KB}'$. In order to perform non-monotonic inferences, as done in [8], we strengthen the semantics of $\mathcal{EL}^{\perp+} \mathbf{T}$ by restricting entailment to a class of minimal (or preferred) models. We call the new logic $\mathcal{EL}^\perp \mathbf{T}_{\min}$. Intuitively, the idea is to restrict our consideration to models that *minimize the non typical instances of a concept*.

Given a KB, we consider a finite set $\mathcal{L}_{\mathbf{T}}$ of concepts: these are the concepts whose non typical instances we want to minimize. We assume that the set $\mathcal{L}_{\mathbf{T}}$ contains at least all concepts C such that $\mathbf{T}(C)$ occurs in the KB or in the query F , where a *query* F is either an assertion $C(a)$ or an inclusion relation $C \sqsubseteq D$. As we have just said, $x \in C^I$ is typical for C if $x \in (\square \neg C)^I$. Minimizing the non typical instances of C therefore means to minimize the objects falsifying $\square \neg C$ for $C \in \mathcal{L}_{\mathbf{T}}$. Hence, for a given model $\mathcal{M} = \langle \Delta, <, I \rangle$, we define:

$$\mathcal{M}_{\mathcal{L}_{\mathbf{T}}}^{\square \neg} = \{(x, \neg \square \neg C) \mid x \notin (\square \neg C)^I, \text{ with } x \in \Delta, C \in \mathcal{L}_{\mathbf{T}}\}.$$

Definition 3 (Preferred and minimal models). Given a model $\mathcal{M} = \langle \Delta, <, I \rangle$ of a knowledge base KB, and a model $\mathcal{M}' = \langle \Delta', <', I' \rangle$ of KB, we say that \mathcal{M} is preferred to \mathcal{M}' w.r.t. $\mathcal{L}_{\mathbf{T}}$, and we write $\mathcal{M} <_{\mathcal{L}_{\mathbf{T}}} \mathcal{M}'$, if (i) $\Delta = \Delta'$, (ii) $\mathcal{M}_{\mathcal{L}_{\mathbf{T}}}^{\square \neg} \subset \mathcal{M}'_{\mathcal{L}_{\mathbf{T}}}^{\square \neg}$, (iii) $a^I = a'^I$ for all $a \in \mathcal{O}$. \mathcal{M} is a minimal model for KB (w.r.t. $\mathcal{L}_{\mathbf{T}}$) if it is a model of KB and there is no other model \mathcal{M}' of KB such that $\mathcal{M}' <_{\mathcal{L}_{\mathbf{T}}} \mathcal{M}$.

Definition 4 (Minimal Entailment in $\mathcal{EL}^\perp \mathbf{T}_{\min}$). A query F is minimally entailed in $\mathcal{EL}^\perp \mathbf{T}_{\min}$ by KB with respect to $\mathcal{L}_{\mathbf{T}}$ if F is satisfied in all models of KB that are minimal with respect to $\mathcal{L}_{\mathbf{T}}$. We write $\text{KB} \models_{\mathcal{EL}^\perp \mathbf{T}_{\min}} F$.

Example 1. The KB of the Introduction can be reformulated as follows in $\mathcal{EL}^{+\perp}\mathbf{T}$: $TaxPayer \sqcap NotTaxPayer \sqsubseteq \perp$; $Parent \sqsubseteq \exists HasChild.\top$; $\exists HasChild.\top \sqsubseteq Parent$; $\mathbf{T}(Student) \sqsubseteq NotTaxPayer$; $\mathbf{T}(Student \sqcap Worker) \sqsubseteq TaxPayer$; $\mathbf{T}(Student \sqcap Worker \sqcap Parent) \sqsubseteq NotTaxPayer$. Let $\mathcal{L}_{\mathbf{T}} = \{Student, Student \sqcap Worker, Student \sqcap Worker \sqcap Parent\}$. We have that $TBox \cup \{Student(john)\} \models_{\mathcal{EL}^{+\perp}\mathbf{T}_{min}} NotTaxPayer(john)$, since $john^I \in (Student \sqcap \neg Student)^I$ for all minimal models $\mathcal{M} = \langle \Delta, <, I \rangle$ of the KB. In contrast, by the non-monotonic character of minimal entailment, $TBox \cup \{Student(john), Worker(john)\} \models_{\mathcal{EL}^{+\perp}\mathbf{T}_{min}} TaxPayer(john)$. Last, notice that $TBox \cup \{\exists HasChild.(Student \sqcap Worker)(jack)\} \models_{\mathcal{EL}^{+\perp}\mathbf{T}_{min}} \exists HasChild.TaxPayer(jack)$. The latter shows that minimal consequence applies to *implicit individuals* as well, without any ad-hoc mechanism.

Theorem 1 (Complexity for $\mathcal{EL}^{+\perp}\mathbf{T}_{min}$ KBs (Theorem 3.1 in [11])). *The problem of deciding whether $KB \models_{\mathcal{EL}^{+\perp}\mathbf{T}_{min}} F$ is EXPTIME-hard.*

To lower the complexity of minimal entailment in $\mathcal{EL}^{+\perp}\mathbf{T}_{min}$, we consider *Left Local* KBs, a restriction similar to that introduced in [3] for circumscribed $\mathcal{EL}^{+\perp}$ KBs.

Definition 5 (Left Local knowledge base). *A Left Local KB only contains subsumptions $C_L^{LL} \sqsubseteq C_R$, where C and C_R are as in Definition 1 and:*

$$C_L^{LL} := C \mid C_L^{LL} \sqcap C_L^{LL} \mid \exists R.\top \mid \mathbf{T}(C)$$

There is no restriction on the ABox.

Observe that the KB in the Example 1 is Left Local, as no concept of the form $\exists R.C$ with $C \neq \top$ occurs on the left hand side of inclusions. In [11] an upper bound for the complexity of $\mathcal{EL}^{+\perp}\mathbf{T}_{min}$ Left Local KBs is provided by a small model theorem. Intuitively, what allows us to keep the size of the small model polynomial is that we reuse the same world to verify the same existential concept throughout the model. This allows us to conclude that:

Theorem 2 (Complexity for $\mathcal{EL}^{+\perp}\mathbf{T}_{min}$ Left Local KBs (Theorem 3.12 in [11])). *If KB is Left Local, the problem of deciding whether $KB \models_{\mathcal{EL}^{+\perp}\mathbf{T}_{min}} F$ is in Π_2^p .*

3 The Logic $DL\text{-}Lite_c\mathbf{T}_{min}$

In this section, we present the extension of the logic $DL\text{-}Lite_{core}$ [5] with the \mathbf{T} operator. We call it $DL\text{-}Lite_c\mathbf{T}_{min}$. The language of $DL\text{-}Lite_c\mathbf{T}_{min}$ is defined as follows.

Definition 6. *We consider an alphabet of concept names \mathcal{C} , of role names \mathcal{R} , and of individuals \mathcal{O} . Given $A \in \mathcal{C}$ and $r \in \mathcal{R}$, we define*

$$C_L := A \mid \exists R.\top \mid \mathbf{T}(A) \quad R := r \mid r^- \quad C_R := A \mid \neg A \mid \exists R.\top \mid \neg \exists R.\top$$

A $DL\text{-}Lite_c\mathbf{T}_{min}$ KB is a pair $(TBox, ABox)$. $TBox$ contains a finite set of concept inclusions of the form $C_L \sqsubseteq C_R$. $ABox$ contains assertions of the form $C(a)$ and $r(a, b)$, where C is a concept C_L or C_R , $r \in \mathcal{R}$, and $a, b \in \mathcal{O}$.

As for $\mathcal{EL}^\perp\mathbf{T}_{min}$, a model \mathcal{M} for $DL\text{-Lite}_c\mathbf{T}_{min}$ is any structure $\langle \Delta, <, I \rangle$, defined as in Definition 2, where I is extended to take care of inverse roles: given $r \in \mathcal{R}$, $(r^-)^I = \{(a, b) \mid (b, a) \in r^I\}$.

In [11] it has been shown that a small model construction similar to the one for Left Local $\mathcal{EL}^\perp\mathbf{T}_{min}$ KBs can be made also for $DL\text{-Lite}_c\mathbf{T}_{min}$. As a difference, in this case, we exploit the fact that, for each atomic role r , the same element of the domain can be used to satisfy all occurrences of the existential $\exists r. \top$. Also, the same element of the domain can be used to satisfy all occurrences of the existential $\exists r^- . \top$.

Theorem 3 (Complexity for $DL\text{-Lite}_c\mathbf{T}_{min}$ KBs (Theorem 4.6 in [11])). *The problem of deciding whether $\text{KB} \models_{DL\text{-Lite}_c\mathbf{T}_{min}} F$ is in Π_2^p .*

4 The Tableau Calculus for Left Local $\mathcal{EL}^\perp\mathbf{T}_{min}$

In this section we present a tableau calculus $\mathcal{TAB}_{min}^{\mathcal{EL}^\perp\mathbf{T}}$ for deciding whether a query F is minimally entailed from a Left Local knowledge base in the logic $\mathcal{EL}^\perp\mathbf{T}_{min}$. It performs a two-phase computation: in the first phase, a tableau calculus, called $\mathcal{TAB}_{PH1}^{\mathcal{EL}^\perp\mathbf{T}}$, simply verifies whether $\text{KB} \cup \{\neg F\}$ is satisfiable in an $\mathcal{EL}^\perp\mathbf{T}$ model, building candidate models; in the second phase another tableau calculus, called $\mathcal{TAB}_{PH2}^{\mathcal{EL}^\perp\mathbf{T}}$, checks whether the candidate models found in the first phase are *minimal* models of KB, i.e. for each open branch of the first phase, $\mathcal{TAB}_{PH2}^{\mathcal{EL}^\perp\mathbf{T}}$ tries to build a model of KB which is preferred to the candidate model w.r.t. Definition 3. The whole procedure $\mathcal{TAB}_{min}^{\mathcal{EL}^\perp\mathbf{T}}$ is formally defined at the end of this section (Definition 7).

The calculus $\mathcal{TAB}_{min}^{\mathcal{EL}^\perp\mathbf{T}}$ tries to build an open branch representing a minimal model satisfying $\text{KB} \cup \{\neg F\}$. The negation of a query $\neg F$ is defined as follows: if $F \equiv C(a)$, then $\neg F \equiv (\neg C)(a)$; if $F \equiv C \sqsubseteq D$, then $\neg F \equiv (C \sqcap \neg D)(x)$, where x does not occur in KB. Notice that we introduce the connective \neg in a very “localized” way. This is very different from introducing the negation all over the knowledge base, and indeed it does not imply that we jump out of the language of $\mathcal{EL}^\perp\mathbf{T}_{min}$.

$\mathcal{TAB}_{min}^{\mathcal{EL}^\perp\mathbf{T}}$ makes use of labels, which are denoted with x, y, z, \dots . Labels represent individuals either named in the ABox or implicitly expressed by existential restrictions. These labels occur in *constraints* (or *labelled* formulas), that can have the form $x \xrightarrow{R} y$ or $x : C$, where x, y are labels, R is a role and C is either a concept or the negation of a concept of $\mathcal{EL}^\perp\mathbf{T}_{min}$ or has the form $\square \neg D$ or $\neg \square \neg D$, where D is a concept.

Let us now analyze the two components of $\mathcal{TAB}_{min}^{\mathcal{EL}^\perp\mathbf{T}}$, starting with $\mathcal{TAB}_{PH1}^{\mathcal{EL}^\perp\mathbf{T}}$.

4.1 The Tableaux Calculus $\mathcal{TAB}_{PH1}^{\mathcal{EL}^\perp\mathbf{T}}$

A tableau of $\mathcal{TAB}_{PH1}^{\mathcal{EL}^\perp\mathbf{T}}$ is a tree whose nodes are tuples $\langle S \mid U \mid W \rangle$. S is a set of constraints, whereas U contains formulas of the form $C \sqsubseteq D^L$, representing subsumption relations $C \sqsubseteq D$ of the TBox. L is a list of labels, used in order to ensure the termination of the tableau calculus. W is a set of labels x_C used in order to build a “small” model, matching the construction of Theorem 3.11 in [11]. A branch is a sequence of nodes $\langle S_1 \mid U_1 \mid W_1 \rangle, \langle S_2 \mid U_2 \mid W_2 \rangle, \dots, \langle S_n \mid U_n \mid W_n \rangle \dots$, where each node $\langle S_i \mid U_i \mid W_i \rangle$ is obtained from its immediate predecessor $\langle S_{i-1} \mid U_{i-1} \mid W_{i-1} \rangle$

by applying a rule of $\mathcal{TAB}_{PH1}^{\mathcal{EL}^{\perp}\mathbf{T}}$, having $\langle S_{i-1} \mid U_{i-1} \mid W_{i-1} \rangle$ as the premise and $\langle S_i \mid U_i \mid W_i \rangle$ as one of its conclusions. A branch is closed if one of its nodes is an instance of a (Clash) axiom, otherwise it is open. A tableau is closed if all its branches are closed. The rules of $\mathcal{TAB}_{PH1}^{\mathcal{EL}^{\perp}\mathbf{T}}$ are presented in Fig. 1. Rules (\exists_1^+) and (\Box^-) are called *dynamic* since they can introduce a new variable in their conclusions. The other rules are called *static*. We do not need any extra rule for the positive occurrences of \Box , since these are taken into account by the computation of $S_{x \rightarrow y}^M$ of (\Box^-) . The (*cut*) rule ensures that, given any concept $C \in \mathcal{L}_{\mathbf{T}}$, an open branch built by $\mathcal{TAB}_{PH1}^{\mathcal{EL}^{\perp}\mathbf{T}}$ contains either $x : \Box \neg C$ or $x : \neg \Box \neg C$ for each label x : this is needed in order to allow $\mathcal{TAB}_{PH2}^{\mathcal{EL}^{\perp}\mathbf{T}}$ to check the minimality of the model corresponding to the open branch. As mentioned above, given a node $\langle S \mid U \mid W \rangle$, each formula $C \sqsubseteq D$ in U is equipped with the list L of labels to which unfolding of the subsumption has already been applied. This avoids multiple unfolding of the same subsumption with the same label.

The calculus $\mathcal{TAB}_{PH1}^{\mathcal{EL}^{\perp}\mathbf{T}}$ is different from the calculus $\mathcal{ALC} + \mathbf{T}_{min}$ [8] in two respects. First, the rule (\exists^+) is split in the two rules $(\exists^+)_1$ and $(\exists^+)_2$. When the rule $(\exists^+)_1$ is applied to a formula $u : \exists R.C$, it introduces a new label x_C only when the set W does not already contain x_C . Otherwise, x_C is already on the branch and $u \xrightarrow{R} x_C$ is simply added to the conclusion of the rule. As a consequence, in a given branch, $(\exists^+)_1$ introduces a unique new label x_C for each concept C occurring in the initial KB in some $\exists R.C$, and no blocking machinery is needed to ensure termination. This simplification is possible since we are considering Left Local KBs, which have small models; in these models all existentials $\exists R.C$ occurring in KB are made true by reusing a single witness x_C (Theorem 3.12 in [11]). Notice also that the rules $(\exists^+)_1$ and $(\exists^+)_2$ introduce a branching on the choice of the label used to realize the existential restriction $u : \exists R.C$. However, just the leftmost conclusion of $(\exists^+)_1$ introduces a new label x_C ; in all the other branches, a label y_i occurring in S is chosen.

Second, in order to build multilinear models of Definition 2, the calculus adopts a strengthened version of the rule (\Box^-) used in $\mathcal{TAB}_{min}^{\mathcal{ALC}+\mathbf{T}}$ [8]. We write \bar{S} as an abbreviation for $S, u : \neg \Box \neg C_1, \dots, u : \neg \Box \neg C_n$. Moreover, we define $S_{u \rightarrow y}^M = \{y : \neg D, y : \Box \neg D \mid u : \Box \neg D \in S\}$ and, for $k = 1, 2, \dots, n$, we define $\bar{S}_{u \rightarrow y}^{\Box^-k} = \{y : \neg \Box \neg C_j \sqcup C_j \mid u : \neg \Box \neg C_j \in \bar{S} \wedge j \neq k\}$. The strengthened rule (\Box^-) contains: (i) n branches, one for each $u : \neg \Box \neg C_k$ in \bar{S} , in which a *new* typical C_k individual x is introduced (i.e. $x : C_k$ and $x : \Box \neg C_k$ are added), and for all other $u : \neg \Box \neg C_j$, either $x : C_j$ holds or the formula $x : \neg \Box \neg C_j$ is recorded; (ii) other $n \times m$ branches, one for each label y_i and for each $u : \neg \Box \neg C_k$ in \bar{S} (m is the number of labels occurring in S): in these branches, a given y_i is chosen as a typical instance of C_k , that is to say $y_i : C_k$ and $y_i : \Box \neg C_k$ are added, and for all other $u : \neg \Box \neg C_j$, either $y_i : C_j$ holds or the formula $y_i : \neg \Box \neg C_j$ is recorded. This rule is sound with respect to multilinear models. The advantage of this rule over the (\Box^-) rule in the calculus $\mathcal{TAB}_{min}^{\mathcal{ALC}+\mathbf{T}}$ is that all the negated box formulas labelled by u are treated in one step, introducing only a new label x in one of the conclusions. To keep \bar{S} readable, we have used \sqcup . Hence, our calculus requires the rule for \sqcup , even if this constructor does not belong to $\mathcal{EL}^{\perp}\mathbf{T}_{min}$.

In order to check the satisfiability of a KB, we build its *corresponding constraint system* $\langle S \mid U \mid \emptyset \rangle$, and we check its satisfiability. Given $\text{KB}=(\text{TBox}, \text{ABox})$, its *corre-*

$\langle S, x : C, x : \neg C \mid U \mid W \rangle$ (Clash)	$\langle S, x : \neg \top \mid U \mid W \rangle$ (Clash) $_{\neg\top}$	$\langle S, x : \perp \mid U \mid W \rangle$ (Clash) $_{\perp}$
$\frac{\langle S, x : C \sqcap D \mid U \mid W \rangle}{\langle S, x : C, x : D \mid U \mid W \rangle}$ (\sqcap^+)	$\frac{\langle S, x : \neg(C \sqcap D) \mid U \mid W \rangle}{\langle S, x : \neg C \mid U \mid W \rangle \langle S, x : \neg D \mid U \mid W \rangle}$ (\sqcap^-)	$\frac{\langle S, x : C \sqcup D \mid U \mid W \rangle}{\langle S, x : C \mid U \mid W \rangle \langle S, x : D \mid U \mid W \rangle}$ (\sqcup^+)
$\frac{\langle S, x : \mathbf{T}(C) \mid U \mid W \rangle}{\langle S, x : C, x : \square \neg C \mid U \mid W \rangle}$ (\mathbf{T}^+)	$\frac{\langle S, x : \neg \mathbf{T}(C) \mid U \mid W \rangle}{\langle S, x : \neg C \mid U \mid W \rangle \langle S, x : \neg \square \neg C \mid U \mid W \rangle}$ (\mathbf{T}^-)	$\frac{\langle S \mid U, C \sqsubseteq D^L \mid W \rangle}{\langle S, x : \neg C \sqcup D \mid U, C \sqsubseteq D^{L,x} \mid W \rangle}$ (Unfold) if x occurs in S and $x \notin L$
$\frac{\langle S, u : \exists R.C \mid U \mid W \rangle}{\langle S, u \xrightarrow{R} x_C, x_C : C \mid U \mid W \cup \{x_C\} \rangle \langle S, u \xrightarrow{R} y_1, y_1 : C \mid U \mid W \rangle \cdots \langle S, u \xrightarrow{R} y_m, y_m : C \mid U \mid W \rangle}$ (\exists^+) $_1$ if $x_C \notin W$ and y_1, \dots, y_m are all the labels occurring in S		
$\frac{\langle S, u : \exists R.C \mid U \mid W \rangle}{\langle S, u \xrightarrow{R} x_C \mid U \mid W \rangle \langle S, u \xrightarrow{R} y_1, y_1 : C \mid U \mid W \rangle \cdots \langle S, u \xrightarrow{R} y_m, y_m : C \mid U \mid W \rangle}$ (\exists^+) $_2$ if $x_C \in W$ and y_1, \dots, y_m are all the labels occurring in S		
$\frac{\langle S, x : \neg \exists R.C, x \xrightarrow{R} y \mid U \mid W \rangle}{\langle S, x : \neg \exists R.C, x \xrightarrow{R} y, y : \neg C \mid U \mid W \rangle}$ (\exists^-) if $y : \neg C \notin S$	$\frac{\langle S \mid U \mid W \rangle}{\langle S, x : \square \neg C \mid U \mid W \rangle \langle S, x : \square \neg C \mid U \mid W \rangle}$ (cut) if $x : \square \neg C \notin S$ and $x : \square \neg C \notin S$ x occurs in S $C \in \mathcal{L}_{\mathbf{T}}$	
$\frac{\langle S, u : \neg \square \neg C_1, u : \neg \square \neg C_2, \dots, u : \neg \square \neg C_n \mid U \mid W \rangle}{\langle S, x : C_k, x : \square \neg C_k, S_{u \rightarrow x}^M, \bar{S}_{u \rightarrow x}^{\square-k} \mid U \mid W \rangle \langle S, y_1 : C_k, y_1 : \square \neg C_k, S_{u \rightarrow y_1}^M, \bar{S}_{u \rightarrow y_1}^{\square-k} \mid U \mid W \rangle \cdots \langle S, y_m : C_k, y_m : \square \neg C_k, S_{u \rightarrow y_m}^M, \bar{S}_{u \rightarrow y_m}^{\square-k} \mid U \mid W \rangle}$ (\square^-) x new if y_1, \dots, y_m are all the labels occurring in $S, y_1 \neq u, \dots, y_m \neq u$ $k = 1, 2, \dots, n$		

Fig. 1. The calculus $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$.

sponding constraint system $\langle S \mid U \mid \emptyset \rangle$ is defined as follows: $S = \{a : C \mid C(a) \in ABox\} \cup \{a \xrightarrow{R} b \mid R(a, b) \in ABox\}$; $U = \{C \sqsubseteq D^\emptyset \mid C \sqsubseteq D \in TBox\}$. KB is satisfiable if and only if its corresponding constraint system $\langle S \mid U \mid \emptyset \rangle$ is satisfiable. In order to verify the satisfiability of $\text{KB} \cup \{\neg F\}$, we use $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ to check the satisfiability of the constraint system $\langle S \mid U \mid \emptyset \rangle$ obtained by adding the constraint corresponding to $\neg F$ to S' , where $\langle S' \mid U \mid \emptyset \rangle$ is the corresponding constraint system of KB. To this purpose, the rules of the calculus $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ are applied until either a contradiction is generated (Clash) or a model satisfying $\langle S \mid U \mid \emptyset \rangle$ can be obtained from the resulting constraint system.

The rules of $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ are applied with the following *standard strategy*: 1. apply a rule to a label x only if no rule is applicable to a label y such that $y \prec x$ (where $y \prec x$ says that label x has been introduced in the tableaux later than y); 2. apply dynamic rules only if no static rule is applicable. In [9] it has been shown that the calculus is sound and complete and terminating. In particular, any tableau generated by $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ for $\langle S \mid U \mid \emptyset \rangle$ is finite, and the length of the tableau branches built by the strategy is $O(n^2)$. This follows from the fact that dynamic rules (\exists^+) $_1$ and (\square^-) generate at most $O(n)$ labels in a branch, and that, for each label, static rules are applied at most $O(n)$ times. Hence, given a KB and a query F , the problem of checking whether $\text{KB} \cup \{\neg F\}$ in $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ is satisfiable is in NP.

$\langle S, x : C, x : \neg C \mid U \mid K \rangle$ (Clash)	$\langle S, x : \neg \top \mid U \mid K \rangle$ (Clash) $_{\neg\top}$	$\langle S, x : \perp \mid U \mid K \rangle$ (Clash) $_{\perp}$
$\langle S \mid U \mid \emptyset \rangle$ (Clash) $_{\emptyset}$	$\langle S, x : \neg \Box \neg C \mid U \mid K \rangle$ (Clash) $_{\Box \neg}$ if $x : \neg \Box \neg C \notin K$	$\frac{\langle S \mid U, C \sqsubseteq D^L \mid K \rangle}{\langle S, x : \neg C \sqcup D \mid U, C \sqsubseteq D^L, x \mid K \rangle}$ (Unfold) $x \in \mathcal{D}(\mathbf{B})$ and $x \notin L$
$\frac{\langle S, x : C \sqcap D \mid U \mid K \rangle}{\langle S, x : C, x : D \mid U \mid K \rangle}$ (\sqcap^+)	$\frac{\langle S, x : \neg(C \sqcap D) \mid U \mid K \rangle}{\langle S, x : \neg C \mid U \mid K \rangle \quad \langle S, x : \neg D \mid U \mid K \rangle}$ (\sqcap^-)	$\frac{\langle S, x : \mathbf{T}(C) \mid U \mid K \rangle}{\langle S, x : C, x : \Box \neg C \mid U \mid K \rangle}$ (\mathbf{T}^+)
$\frac{\langle S, x : \neg \mathbf{T}(C) \mid U \mid K \rangle}{\langle S, x : \neg C \mid U \mid K \rangle \quad \langle S, x : \neg \Box \neg C \mid U \mid K \rangle}$ (\mathbf{T}^-)	$\frac{\langle S \mid U \mid K \rangle}{\langle S, x : \Box \neg C \mid U \mid K \rangle \quad \langle S, x : \neg \Box \neg C \mid U \mid K \rangle}$ (cut) if $x : \neg \Box \neg C \notin S$ and $x : \Box \neg C \notin S$ $x \in \mathcal{D}(\mathbf{B}) \quad C \in \mathcal{L}_{\mathbf{T}}$	
$\frac{\langle S, u : \exists R.C \mid U \mid K \rangle}{\langle S, u \xrightarrow{R} y_1, y_1 : C \mid U \mid K \rangle \quad \cdots \quad \langle S, u \xrightarrow{R} y_m, y_m : C \mid U \mid K \rangle}$ (\exists^+) if $\mathcal{D}(\mathbf{B}) = \{y_1, \dots, y_m\}$		
$\frac{\langle S, u : \neg \Box \neg C_1, \dots, u : \neg \Box \neg C_n \mid U \mid K, u : \neg \Box \neg C_1, \dots, u : \neg \Box \neg C_n \rangle}{\langle S, y_1 : C_k, y_1 : \Box \neg C_k, S_{u \rightarrow y_1}^M, \bar{S}_{u \rightarrow y_1}^{\Box \neg k} \mid U \mid K \rangle \quad \cdots \quad \langle S, y_m : C_k, y_m : \Box \neg C_k, S_{u \rightarrow y_m}^M, \bar{S}_{u \rightarrow y_m}^{\Box \neg k} \mid U \mid K \rangle}$ (\Box^-) if $\mathcal{D}(\mathbf{B}) = \{y_1, \dots, y_m\}$ and $y_1 \neq u, \dots, y_m \neq u$		

Fig. 2. The calculus $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$. To save space, we omit the rule (\sqcup^+).

4.2 The Tableaux Calculus $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$

Let us now introduce the calculus $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ which, for each open branch \mathbf{B} built by $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$, verifies whether it represents a minimal model of the KB. Given an open branch \mathbf{B} of a tableau built from $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$, let $\mathcal{D}(\mathbf{B})$ be the set of labels occurring on \mathbf{B} . Moreover, let \mathbf{B}^{\Box^-} be the set of formulas $x : \neg \Box \neg C$ occurring in \mathbf{B} , that is to say $\mathbf{B}^{\Box^-} = \{x : \neg \Box \neg C \mid x : \neg \Box \neg C \text{ occurs in } \mathbf{B}\}$.

A tableau of $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ is a tree whose nodes are tuples of the form $\langle S \mid U \mid K \rangle$, where S and U are defined as in a constraint system, whereas K contains formulas of the form $x : \neg \Box \neg C$, with $C \in \mathcal{L}_{\mathbf{T}}$. The basic idea of $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ is as follows. Given an open branch \mathbf{B} built by $\mathcal{TAB}_{PH1}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ and corresponding to a model $\mathcal{M}^{\mathbf{B}}$ of $\text{KB} \cup \{\neg F\}$, $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ checks whether $\mathcal{M}^{\mathbf{B}}$ is a minimal model of KB by trying to build a model of KB which is preferred to $\mathcal{M}^{\mathbf{B}}$. To this purpose, it keeps track (in K) of the negated box used in \mathbf{B} (\mathbf{B}^{\Box^-}) in order to check whether it is possible to build a model of KB containing less negated box formulas. The tableau built by $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ closes if it is not possible to build a model smaller than $\mathcal{M}^{\mathbf{B}}$, it remains open otherwise. Since by Definition 3 two models can be compared only if they have the same domain, $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ tries to build an open branch containing all the labels appearing on \mathbf{B} , i.e. those in $\mathcal{D}(\mathbf{B})$. To this aim, the dynamic rules use labels in $\mathcal{D}(\mathbf{B})$ instead of introducing new ones in their conclusions. The rules of $\mathcal{TAB}_{PH2}^{\mathcal{E}\mathcal{L}^{\perp}\mathbf{T}}$ are shown in Fig. 2.

More in detail, the rule (\exists^+), when applied to a formula $x : \exists R.C$, introduces, for each label $y \in \mathcal{D}(\mathbf{B})$, $x \xrightarrow{R} y$ and $y : C$. The choice of the label y introduces a branching in the tableau construction. The rule (Unfold) is applied to *all the labels* of $\mathcal{D}(\mathbf{B})$ (and not only to those appearing in the branch). The rule (\Box^-) is applied to a node $\langle S, u : \neg \Box \neg C_1, \dots, u : \neg \Box \neg C_n \mid U \mid K \rangle$, when $\{u : \neg \Box \neg C_1, \dots, u :$

$\neg\Box\neg C_n\} \subseteq K$, i.e. when the negated box formulas $u : \neg\Box\neg C_i$ also belong to the open branch \mathbf{B} . Also in this case, the rule introduces a branch on the choice of the individual $y_i \in \mathcal{D}(\mathbf{B})$ to be used in the conclusion. In case a tableau node has the form $\langle S, x : \neg\Box\neg C \mid U \mid K \rangle$, and $x : \neg\Box\neg C \notin K$, then $\mathcal{TAB}_{PH2}^{\mathcal{EL}^+\mathbf{T}}$ detects a clash, called $(\text{Clash})_{\Box^-}$: this corresponds to the situation where $x : \neg\Box\neg C$ does not belong to \mathbf{B} , while the model corresponding to the branch being built contains $x : \neg\Box\neg C$, and hence is *not* preferred to the model represented by \mathbf{B} .

The calculus $\mathcal{TAB}_{PH2}^{\mathcal{EL}^+\mathbf{T}}$ also contains the clash condition $(\text{Clash})_{\emptyset}$. Since each application of (\Box^-) removes the negated box formulas $x : \neg\Box\neg C_i$ from the set K , when K is empty all the negated boxed formulas occurring in \mathbf{B} also belong to the current branch. In this case, the model built by $\mathcal{TAB}_{PH2}^{\mathcal{EL}^+\mathbf{T}}$ satisfies the same set of $x : \neg\Box\neg C_i$ (for all individuals) as \mathbf{B} and, thus, it is not preferred to the one represented by \mathbf{B} .

Let \mathbf{KB} be a knowledge base whose corresponding constraint system is $\langle S \mid U \mid \emptyset \rangle$. Let F be a query and let S' be the set of constraints obtained by adding to S the constraint corresponding to $\neg F$. $\mathcal{TAB}_{PH2}^{\mathcal{EL}^+\mathbf{T}}$ is *sound and complete* in the following sense: an open branch \mathbf{B} built by $\mathcal{TAB}_{PH1}^{\mathcal{EL}^+\mathbf{T}}$ for $\langle S' \mid U \mid \emptyset \rangle$ is satisfiable in a minimal model of \mathbf{KB} iff the tableau in $\mathcal{TAB}_{PH2}^{\mathcal{EL}^+\mathbf{T}}$ for $\langle S \mid U \mid \mathbf{B}^{\Box^-} \rangle$ is closed.

Termination of the calculus $\mathcal{TAB}_{PH2}^{\mathcal{EL}^+\mathbf{T}}$ is ensured by the fact that dynamic rules make use of labels belonging to $\mathcal{D}(\mathbf{B})$, which is finite, rather than introducing “new” labels in the tableau. Also, it is possible to show that the problem of verifying that a branch \mathbf{B} represents a minimal model for \mathbf{KB} in $\mathcal{TAB}_{PH2}^{\mathcal{EL}^+\mathbf{T}}$ is in NP in the size of \mathbf{B} .

The overall procedure $\mathcal{TAB}_{min}^{\mathcal{ALC}^+\mathbf{T}}$ is defined as follows:

Definition 7. *Let \mathbf{KB} be a knowledge base whose corresponding constraint system is $\langle S \mid U \mid \emptyset \rangle$. Let F be a query and let S' be the set of constraints obtained by adding to S the constraint corresponding to $\neg F$. The calculus $\mathcal{TAB}_{min}^{\mathcal{EL}^+\mathbf{T}}$ checks whether a query F is minimally entailed from \mathbf{KB} by means of the following procedure: (phase 1) the calculus $\mathcal{TAB}_{PH1}^{\mathcal{EL}^+\mathbf{T}}$ is applied to $\langle S' \mid U \mid \emptyset \rangle$; if, for each branch \mathbf{B} built by $\mathcal{TAB}_{PH1}^{\mathcal{EL}^+\mathbf{T}}$, either (i) \mathbf{B} is closed or (ii) (phase 2) the tableau built by the calculus $\mathcal{TAB}_{PH2}^{\mathcal{EL}^+\mathbf{T}}$ for $\langle S \mid U \mid \mathbf{B}^{\Box^-} \rangle$ is open, then $\mathbf{KB} \models_{min}^{\mathcal{L}\tau} F$, otherwise $\mathbf{KB} \not\models_{min}^{\mathcal{L}\tau} F$.*

In [9] it has been shown that $\mathcal{TAB}_{min}^{\mathcal{EL}^+\mathbf{T}}$ is a sound and complete decision procedure for verifying if $\mathbf{KB} \models_{\mathcal{EL}^+\mathbf{T}_{min}} F$. Furthermore, the problem of deciding whether $\mathbf{KB} \models_{\mathcal{EL}^+\mathbf{T}_{min}} F$ by means of $\mathcal{TAB}_{min}^{\mathcal{EL}^+\mathbf{T}}$ is in Π_2^p .

5 A Tableau Calculus for $DL\text{-Lite}_c\mathbf{T}_{min}$

In this section we shortly describe a tableau calculus $\mathcal{TAB}_{min}^{Lite_c\mathbf{T}}$ for deciding query entailment in the logic $DL\text{-Lite}_c\mathbf{T}_{min}$. The calculus is similar to the one introduced for $\mathcal{EL}^+\mathbf{T}_{min}$ in the previous section, however it is significantly different from it in the definition of some of the rules. Given a set of constraints S and a role $r \in \mathcal{R}$, let $r(S) = \{x \xrightarrow{r} y \mid x \xrightarrow{r} y \in S\}$. The calculus $\mathcal{TAB}_{PH1}^{Lite_c\mathbf{T}}$ used in the first phase differs from $\mathcal{TAB}_{PH1}^{\mathcal{EL}^+\mathbf{T}}$ in the following points:

1. As in the calculus $\mathcal{TAB}_{PH1}^{\mathcal{EL}^{\perp}\mathbf{T}}$, the split of the (\exists^+) in the two rules:

$\frac{\langle S, x : \exists r. \top \mid U \rangle}{\langle S, x \xrightarrow{r} y \mid U \rangle \langle S, x \xrightarrow{r} y_1 \mid U \rangle \dots \langle S, x \xrightarrow{r} y_m \mid U \rangle} (\exists^+)_1^+$ <p style="text-align: center; margin: 0;">if $r(S) = \emptyset$ if y_1, \dots, y_m are all the labels occurring in S</p>	$\frac{\langle S, x : \exists r. \top \mid U \rangle}{\langle S, x \xrightarrow{r} y_1 \mid U \rangle \dots \langle S, x \xrightarrow{r} y_m \mid U \rangle} (\exists^+)_2^+$ <p style="text-align: center; margin: 0;">if $r(S) \neq \emptyset$ if y_1, \dots, y_m are all the labels occurring in S</p>
---	--

reflects the main idea of the construction of a small model at the base of Theorem 4.5 in [11]. Such small model theorem essentially shows that $DL\text{-}Lite_c\mathbf{T}_{min}$ KBs can have small models in which all existentials $\exists R. \top$ occurring in KB are made true in the model by reusing a single witness y . In the calculus we use the same idea: when the rule $(\exists^+)_1^+$ is applied to a formula $x : \exists r. \top$, it introduces a new label y and the constraint $x \xrightarrow{r} y$ only when there is no other previous constraint $u \xrightarrow{r} v$ in S , i.e. $r(S) = \emptyset$. Otherwise, rule $(\exists^+)_2^+$ is applied and it introduces $x \xrightarrow{r} y$. As a consequence, $(\exists^+)_2^+$ does not introduce any new label in the branch whereas $(\exists^+)_1^+$ only introduces a new label y for each role r occurring in the initial KB in some $\exists r. \top$ and no blocking machinery is needed to ensure termination.

2. In order to keep into account inverse roles, two further rules for existential formulas are introduced:

$\frac{\langle S, x : \exists r^-. \top \mid U \rangle}{\langle S, y \xrightarrow{r} x \mid U \rangle \langle S, y_1 \xrightarrow{r} x \mid U \rangle \dots \langle S, y_m \xrightarrow{r} x \mid U \rangle} (\exists^+)_1^-$ <p style="text-align: center; margin: 0;">if $r(S) = \emptyset$ if y_1, \dots, y_m are all the labels occurring in S</p>	$\frac{\langle S, x : \exists r^-. \top \mid U \rangle}{\langle S, y_1 \xrightarrow{r} x \mid U \rangle \dots \langle S, y_m \xrightarrow{r} x \mid U \rangle} (\exists^+)_2^-$ <p style="text-align: center; margin: 0;">if $r(S) \neq \emptyset$ if y_1, \dots, y_m are all the labels occurring in S</p>
---	--

These rules work similarly to $(\exists^+)_1^+$ and $(\exists^+)_2^+$ in order to build a branch representing a small model: when the rule $(\exists^+)_1^-$ is applied to a formula $x : \exists r^-. \top$, it introduces a new label y and the constraint $y \xrightarrow{r} x$ only when there is no other constraint $u \xrightarrow{r} v$ in S . Otherwise, since a constraint $y \xrightarrow{r} u$ has been already introduced in that branch, $y \xrightarrow{r} x$ is added to the conclusion of the rule.

3. Negated existential formulas can occur in a branch, but only having the form (i) $x : \neg \exists r. \top$ or (ii) $x : \neg \exists r^-. \top$. (i) means that x has no relationships with other individuals via the role r , i.e. we need to detect a contradiction if both (i) and $x \xrightarrow{r} y$ belong to the same branch (for some y), and mark the branch as closed. The clash condition $(\text{Clash})_r$ is added to the calculus $\mathcal{TAB}_{PH1}^{Lite_c\mathbf{T}}$ in order to detect such a situation. Analogously, (ii) means that there is no y such that y is related to x by means of r , then $(\text{Clash})_{r^-}$ is introduced in order to close a branch containing both (ii) and, for some y , a constraint $y \xrightarrow{r} x$. These clash conditions are as follows:

$\langle S, x \xrightarrow{r} y, x : \neg \exists r. \top \mid U \rangle (\text{Clash})_r$	$\langle S, y \xrightarrow{r} x, x : \neg \exists r^-. \top \mid U \rangle (\text{Clash})_{r^-}$
--	--

Apart from the differences above, the rules of $\mathcal{TAB}_{PH1}^{Lite_c\mathbf{T}}$ are the same as those of $\mathcal{TAB}_{PH1}^{\mathcal{EL}^{\perp}\mathbf{T}}$. Similarly for the calculus $\mathcal{TAB}_{PH2}^{Lite_c\mathbf{T}}$ used in the second phase. In [10] it has been shown that both $\mathcal{TAB}_{PH1}^{Lite_c\mathbf{T}}$ and $\mathcal{TAB}_{PH2}^{Lite_c\mathbf{T}}$ are sound, complete and terminating. Furthermore, the problem of deciding whether $\text{KB} \models_{DL\text{-}Lite_c\mathbf{T}_{min}} F$ by means of $\mathcal{TAB}_{min}^{Lite_c\mathbf{T}}$ is in Π_2^p .

6 Conclusions

We have proposed a non-monotonic extension of low complexity Description Logics \mathcal{EL}^\perp and $DL-Lite_{core}$ for reasoning about typicality and defeasible properties. We have summarized complexity results recently studied for such extensions [11], namely that entailment is EXPTIME-hard for $\mathcal{EL}^\perp \mathbf{T}_{min}$, whereas it drops to Π_2^P when considering the Left Local Fragment of $\mathcal{EL}^\perp \mathbf{T}_{min}$. The same Π_2^P complexity has been found for $DL-Lite_c \mathbf{T}_{min}$. These results match the complexity upper bounds of the same fragments in circumscribed KBs [3]. We have also provided tableau calculi for checking minimal entailment in the Left Local fragment of $\mathcal{EL}^\perp \mathbf{T}_{min}$ as well as in $DL-Lite_c \mathbf{T}_{min}$. The proposed calculi match the complexity results above. Of course, many optimizations are possible and we intend to study them in future work.

As mentioned in the Introduction, several non-monotonic extensions of DLs have been proposed in the literature and we refer to [12] for a survey. Concerning non-monotonic extensions of low complexity DLs, the complexity of *circumscribed* fragments of the \mathcal{EL}^\perp and $DL-Lite$ families have been studied in [3]. Recently, a fragment of \mathcal{EL}^\perp for which the complexity of circumscribed KBs is polynomial has been identified in [14]. In future work, we shall investigate complexity of minimal entailment for such a fragment extended with \mathbf{T} and possibly the definition of a calculus for it.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: IJCAI. pp. 364–369 (2005)
2. Baader, F., Hollunder, B.: Priorities on defaults with prerequisites, and their application in treating specificity in terminological default logic. J. of Autom. Reas. 15(1), 41–68 (1995)
3. Bonatti, P.A., Faella, M., Sauro, L.: Defeasible inclusions in low-complexity DLs. J. Artif. Intell. Res. (JAIR) 42, 719–764 (2011)
4. Bonatti, P.A., Lutz, C., Wolter, F.: The complexity of circumscription in DLs. J. Artif. Intell. Res. (JAIR) 35, 717–773 (2009)
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in Description Logics: the DL-Lite family. J. Autom. Reasoning (JAR) 39(3), 385–429 (2007)
6. Casini, G., Straccia, U.: Rational closure for defeasible DLs. In: JELIA. pp. 77–90 (2010)
7. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Trans. Comput. Log. 3(2), 177–225 (2002)
8. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Reasoning about typicality in preferential Description Logics. In: JELIA. pp. 192–205 (2008)
9. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A tableau calculus for a nonmonotonic extension of \mathcal{EL}^\perp . In: TABLEAUX. pp. 180–195 (2011)
10. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A tableau calculus for a nonmonotonic extension of the Description Logic $DL-Lite_{core}$. In: AI*IA. pp. 164–176 (2011)
11. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Reasoning about typicality in low complexity DLs: the logics $\mathcal{EL}^\perp \mathbf{T}_{min}$ and $DL-lite_c \mathbf{T}_{min}$. In: IJCAI. pp. 894–899 (2011)
12. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.: $\mathcal{ALC} + \mathbf{T}_{min}$: a preferential extension of Description Logics. Fundamenta Informaticae 96, 1–32 (2009)
13. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
14. Bonatti, P.A., Faella, M., Sauro, L.: \mathcal{EL} with default attributes and overriding. In: ISWC. pp. 64–79 (2010)
15. Straccia, U.: Default inheritance reasoning in hybrid kl-one-style logics. In: IJCAI. pp. 676–681 (1993)

Concept-Based Semantic Difference in Expressive Description Logics

Rafael S. Gonçalves, Bijan Parsia, and Ulrike Sattler

School of Computer Science
University of Manchester
Manchester, United Kingdom

Abstract. Detecting, much less understanding, the difference between two description logic based ontologies is challenging for ontology engineers due, in part, to the possibility of complex, non-local logic effects of axiom changes. It is often quite difficult to even determine which terms have had their meaning altered by a change. To address this, various principled notions of “semantic diff” (based on deductive inseparability) have been proposed in the literature and have been shown to be computationally practical for the expressively restricted case of \mathcal{ELH}^r -terminologies (which covers significant fragments of SNOMED-CT). However, problems arise even for such limited logics as \mathcal{ACC} : First, computation gets more difficult, becoming undecidable for logics such as \mathcal{SROIQ} which underly the Web Ontology Language (OWL). Second, the presence of negation and disjunction make the standard semantic difference too sensitive to change: essentially, any logically effectual change always affects all terms in the ontology. To address these issues, we formulate the central notion of finding the *minimal change set* based on model inseparability, and present a method to differentiate changes which are specific to (and “of interest” for) particular concept names. Subsequently we present a series of computable approximations, and compare the variously approximated change sets over a series of versions of the NCI Thesaurus (NCIt).

1 Introduction

Determining the significant differences between two documents (so-called “diff”) is a standard and significant problem across a wide range of activities, notably software development. Standard textual diffing algorithms perform poorly on description logic (DL) based ontologies, both for structural reasons (e.g., ontology serializations, such as those of OWL, tend not to impose stable ordering of axioms), and due to the highly non-local and unintuitive logical effects of changes to axioms. Syntactic diffs, such as those based on OWL’s notion of “structural equivalence” [4, 8, 12], detect axiomatic changes between ontologies, but fall short on the identification of differences w.r.t. their entailment sets. Recent notions of semantic difference based on conservative extensions have provided a robust theoretical and practical basis for analysing these logical effects. In particular, they provide a means for determining which terms have had their meaning “affected” by an edit even if that effect is not readily determined by syntactic analysis.

Unfortunately, semantic difference is computationally expensive even for inexpressive logics such as \mathcal{EL} . For the very expressive logics such as $SR\mathcal{OIQ}$ (the DL underlying OWL 2) it is undecidable [10]. Furthermore, as we discuss in this paper, semantic difference runs into other difficulties in more expressive logics. In particular, if we compare entailment sets over logics with disjunction and negation we easily end up with vacuously altered terms: any logically effectual change will alter the meaning of every term.

In this paper, we provide a non-trivializable notion of semantic difference and a series of computable approximations of it for expressive description logics. We evaluate these algorithms on a select subset of the National Cancer Institute Thesaurus (NCIt) corpus, comparing the changes found via the proposed approximations and related approaches. Our experiments show that one approximation, “Grammar diff”, finds significantly more changes than all other methods across the corpus and far more than are identified in the NCIt change logs.

2 Preliminaries

We assume the reader to be reasonably familiar with ontologies and OWL, as well as the underlying description logics (DLs) [1]. We use *terms* to refer to concept and role names. When comparing two ontologies we refer to them as \mathcal{O}_1 and \mathcal{O}_2 , and their *signatures* (i.e., the set of terms occurring in them) as $\tilde{\mathcal{O}}_1$ and $\tilde{\mathcal{O}}_2$, respectively. The signature of an axiom α is denoted $\tilde{\alpha}$. Throughout this paper we use the standard description and first order logic notion of entailment; an axiom α entailed by an ontology \mathcal{O} is denoted $\mathcal{O} \models \alpha$. We refer to an *effectual* addition (removal) from \mathcal{O}_1 to \mathcal{O}_2 as an axiom α such that $\alpha \in \mathcal{O}_2$ and $\mathcal{O}_1 \not\models \alpha$ ($\alpha \in \mathcal{O}_1$ and $\mathcal{O}_2 \not\models \alpha$) [4]. Thus two ontologies are logically equivalent, denoted $\mathcal{O}_1 \equiv \mathcal{O}_2$, if there is no effectual change (addition or removal) between \mathcal{O}_1 and \mathcal{O}_2 . We also use the notion of a *locality-based module* [2]; a module \mathcal{M} of \mathcal{O} for a set of terms (signature) Σ is a subset of \mathcal{O} that preserves all entailments of \mathcal{O} w.r.t. Σ . A \perp -*module* (\top -*module*) extracted from an ontology \mathcal{O} for Σ is denoted $\perp\text{-mod}(\Sigma, \mathcal{O})$ ($\top\text{-mod}(\Sigma, \mathcal{O})$). The set of *subconcepts* of an ontology \mathcal{O} is recursively defined as all subconcepts found in each axiom of \mathcal{O} , plus $\{\top, \perp\}$.

The restriction of an interpretation \mathcal{I} to a set of terms Σ is denoted $\mathcal{I}|_\Sigma$. Two interpretations \mathcal{I} and \mathcal{J} coincide on a signature Σ (denoted $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$) if $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $t^{\mathcal{I}} = t^{\mathcal{J}}$ for each $t \in \Sigma$.

Throughout this paper we use the notion of model conservative extension (mCE) [3, 10], and associated inseparability relation [14]. The notions of mCE-based inseparability, Σ -difference and Σ -entailment are, respectively:

Definition 1 *Given two ontologies $\mathcal{O}_1, \mathcal{O}_2$ over a DL \mathcal{L} , and a signature Σ .*

- (1) \mathcal{O}_2 is model Σ -inseparable from \mathcal{O}_1 ($\mathcal{O}_1 \equiv_\Sigma^{mCE} \mathcal{O}_2$) w.r.t. \mathcal{L}
if $\{\mathcal{I}|_\Sigma \mid \mathcal{I} \models \mathcal{O}_1\} = \{\mathcal{J}|_\Sigma \mid \mathcal{J} \models \mathcal{O}_2\}$
- (2) $\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma = \{\eta \mid \mathcal{O}_1 \not\models \eta, \mathcal{O}_2 \models \eta \text{ and } \eta \text{ is a GCI over } \mathcal{L},$
with $\tilde{\eta} \subseteq \Sigma\}$
- (3) \mathcal{O}_1 Σ -entails \mathcal{O}_2 if $\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma = \emptyset$

3 State of the Art in Semantic Diff

The tool ContentCVS [6] employs a notion of deductive difference (for OWL 2 ontologies) which takes into account entailments of type $A \sqsubseteq C$,¹ where C is a concept formed over grammar G_{cvs} and A, B are concept names, as follows:

Grammar G_{cvs}

$$C \longrightarrow B \mid \exists r.B \mid \forall r.B \mid \neg B$$

The rationale behind the use of this grammar is not exactly clear, and seems rather ad hoc. In a user study of ContentCVS, users criticised “the excessive amount of information displayed when using larger approximations of the deductive difference” [6]. This suggests that, instead of focusing on presenting entailments in the difference, we might prefer to present which concept names are affected by those entailments, and how (e.g., specialised or generalised).

The diff method underlying the system CEX [7] establishes a way to compute the semantic differences between two ontologies,² based on the notion of Σ -entailment, and corresponding diff notion Σ -difference. The output of CEX is a set of entailed axioms in the Σ -difference, so called *witness axioms*, and associated affected terms (denoted $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$). The set $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ contains specialised (denoted $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma^L$) and generalised ($\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma^R$) concept names, as defined in [7]. The set $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma^L$ contains those concept names A for which there is a witness axiom $\alpha : A \sqsubseteq C$ that follows from \mathcal{O}_2 but not \mathcal{O}_1 . The concept C in such axioms α is called a *witness* for the change in A . In $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma^R$ the witness is the subsumer rather than the subsumee.

The computational complexity of deciding Σ -entailment is undecidable for expressive DLs such as *SRQLQ*. For \mathcal{EL} it is already ExpTime-complete [11], while for \mathcal{ALL} , \mathcal{ALLQ} , and \mathcal{ALLQL} it is 2ExpTime-complete [10]. Aside from the high complexity result, a direct extension of Σ -difference for more expressive logics such as \mathcal{ALL} would fail; when we step beyond \mathcal{EL} as a witness language into more expressive logics with disjunction and negation, then we can create a vacuously true witness that would make $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ contain all terms in Σ (so long as $\mathcal{O}_1 \neq \mathcal{O}_2$). The ontologies need not be in the witness language; in fact consider the following \mathcal{EL} ontologies: $\mathcal{O}_1 = \{A \sqsubseteq B, C \sqsubseteq \top, D \sqsubseteq \top\}$, and $\mathcal{O}_2 = \{A \sqsubseteq B, C \sqsubseteq D\}$. Clearly \mathcal{O}_2 is a conservative extension of \mathcal{O}_1 w.r.t. $\Sigma = \{A, B\}$, but if we take $\Sigma' = \{\widetilde{\mathcal{O}}_1 \cap \widetilde{\mathcal{O}}_2\}$ then that is no longer the case. A witness axiom for the separability would be, e.g., $\eta := A \sqsubseteq \neg C \sqcup D$. This witness “witnesses” a change to every concept $A' \in \Sigma'$; for each witness axiom $\eta' : A' \sqsubseteq \neg C \sqcup D$ we have that $\mathcal{O}_1 \not\models \eta'$, while $\mathcal{O}_2 \models \eta'$. Such a witness would suffice to pinpoint, according to Σ -difference, that all terms in Σ' have changed: $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma'} = \Sigma'$ since $\top \sqsubseteq \neg C \sqcup D$. Consequently, this kind of witnesses are uninteresting for any particular concept aside from \top . Likewise, a change $A \sqsubseteq \perp$ implies that, for all B in the signature of the ontology in question, we have that $A \sqsubseteq B$. Yet these consequences are of no interest to any concept B .

¹ Additionally, ContentCVS also compares role hierarchies.

² Albeit the implementation is restricted to acyclic \mathcal{ELH}^r terminologies (\mathcal{EL} extended with role inclusions and range restrictions).

Similar to the case of the least common subsumer [9], the presence of disjunction (and negation) trivialises definitions that are meaningful in less expressive logics. This phenomenon conveys the need to move to another diff notion when dealing with propositionally closed ontologies, one which distinguishes directly affected terms (thus “specific” changes) and indirectly affected terms (such as those via \top and \perp from previous examples).

4 Semantic Diff

Given the shortcomings of existing methodologies, we present a semantic diff method that *a)* determines which concepts have been affected by changes. For exposition reasons, we concentrate on concepts, though roles are easily added. And *b)* identifies which concepts have been directly (or indirectly) changed.

Ideally, a solution to these problems would be *1)* a computationally feasible function (for OWL 2 ontologies), *2)* based on a principled grammar, that *3)* returns those concept names affected by changes between two ontologies, while *4)* distinguishing whether each concept name is directly (or indirectly) specialised and/or generalised.

4.1 Determining the Change Set

Given two ontologies \mathcal{O}_1 and \mathcal{O}_2 , such that $\mathcal{O}_1 \not\equiv \mathcal{O}_2$ (i.e. there exists at least one effectual change in $\text{Diff}(\mathcal{O}_1, \mathcal{O}_2)$), we know that \mathcal{O}_1 and \mathcal{O}_2 are not Σ -inseparable (for $\Sigma = \tilde{\mathcal{O}}_1 \cup \tilde{\mathcal{O}}_2$ w.r.t. model inseparability, i.e. $\mathcal{O}_1 \not\equiv_{\Sigma}^{mCE} \mathcal{O}_2$ since an effectual change implies some change in semantics. In order to pinpoint this change, we need to find the set of terms Σ' s.t. \mathcal{O}_1 is mCE-inseparable from \mathcal{O}_2 w.r.t. the remaining signature $\Sigma \setminus \Sigma'$: $\mathcal{O}_1 \equiv_{\Sigma \setminus \Sigma'}^{mCE} \mathcal{O}_2$. Then we know that, from \mathcal{O}_1 to \mathcal{O}_2 , there are no changes in entailments over $\Sigma \setminus \Sigma'$. We refer to this set of terms Σ' as the Minimal Change Set (denoted $\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)$), in the sense that we can formulate a non-trivial entailment η over Σ' s.t. $\mathcal{O}_1 \not\models \eta$ but $\mathcal{O}_2 \models \eta$. Thus we denote these terms as *affected*.

Definition 2 (Minimal Affected Terms) *A set $\Sigma' \subseteq \Sigma$ is a set of minimal affected terms between \mathcal{O}_1 and \mathcal{O}_2 if:*

$$\mathcal{O}_1 \not\equiv_{\Sigma'}^{mCE} \mathcal{O}_2 \text{ and for all } \Sigma'' \subsetneq \Sigma' : \mathcal{O}_1 \equiv_{\Sigma''}^{mCE} \mathcal{O}_2.$$

The set of all such sets is denoted $\text{MinAT}(\mathcal{O}_1, \mathcal{O}_2)$.

In order to form the minimal change set, we take the union over all sets of affected terms in $\text{MinAT}(\mathcal{O}_1, \mathcal{O}_2)$.

Definition 3 (Minimal Change Set) *The minimal change set, denoted $\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)$, of two ontologies is defined as follows:*

$$\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2) := \bigcup \text{MinAT}(\mathcal{O}_1, \mathcal{O}_2).$$

Given a set of witness axioms, we can tell apart specialised and generalised concepts depending on whether the witness concept is on the right hand side (RHS) or the left hand side (LHS) of the witness axiom, accordingly. Furthermore, we regard a concept name A as directly specialised (generalised) via some witness C if there is no concept name B that is a superconcept (subconcept) of A , and C is also a witness for a change in B . Otherwise A changed indirectly.

Definition 4 (Affected Terms) *For a diff function Φ , the sets of affected concept names for a signature Σ are:*

$$\begin{aligned}\Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^L &= \{A \in \Sigma \mid \text{there exists } A \sqsubseteq C \in \Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}\} \\ \Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^R &= \{A \in \Sigma \mid \text{there exists } C \sqsubseteq A \in \Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}\} \\ \Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^{\top} &= \begin{cases} \{\top\} & \text{if there is a } \top \sqsubseteq C \in \Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma} \\ \emptyset & \text{otherwise} \end{cases} \\ \Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^{\perp} &= \begin{cases} \{\perp\} & \text{if there is a } C \sqsubseteq \perp \in \Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma} \\ \emptyset & \text{otherwise} \end{cases} \\ \Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^Y &= \bigcup_{Y \in \{L, R, \top, \perp\}} \Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^Y\end{aligned}$$

Given a concept name $A \in \Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^L$ (analogously $A \in \Phi\text{-AT}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^R$), and a set of terms $\Sigma^+ := \Sigma \cup \{\top, \perp\}$:

A direct change of A is a witness C s.t. $A \sqsubseteq C$ ($C \sqsubseteq A$) $\in \Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)$ and there is no $B \in \Sigma^+$ s.t. $\mathcal{O}_2 \models A \sqsubseteq B$ ($\mathcal{O}_2 \models B \sqsubseteq A$), $\mathcal{O}_2 \not\models A \equiv B$, and $B \sqsubseteq C$ ($C \sqsubseteq B$) $\in \Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)$.

An indirect change of A is a witness C s.t. $A \sqsubseteq C$ ($C \sqsubseteq A$) $\in \Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)$ and there is at least one $B \in \Sigma^+$ s.t. $\mathcal{O}_2 \models A \sqsubseteq B$ ($\mathcal{O}_2 \models B \sqsubseteq A$), $\mathcal{O}_2 \not\models A \equiv B$ and $B \sqsubseteq C$ ($C \sqsubseteq B$) $\in \Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)$.

Concept A is purely directly changed if it is only directly changed (analogously for purely indirectly changed).

As an example, given ontologies $\mathcal{O}_1 := \{A \sqsubseteq B, \exists r.C \sqsubseteq D\}$ and $\mathcal{O}_2 := \mathcal{O}_1 \cup \{B \sqsubseteq \exists r.C\}$, we have that B is purely directly specialised via witness $\exists r.C$, while A is indirectly specialised via the same witness, since $\mathcal{O}_2 \models A \sqsubseteq B$ and $B \sqsubseteq \exists r.C \in \text{Diff}(\mathcal{O}_1, \mathcal{O}_2)$, in other words, concept A changes via B .

The distinction between directly- and indirectly-affected concept names, and the separation of concepts affected via \top and \perp , allows us to overcome the problems described in Section 3, w.r.t. propositionally closed description logics.

4.2 Computation

Deciding the minimal change set between two ontologies involves deciding whether, for a given signature Σ , two ontologies are mCE-inseparable w.r.t. Σ . Since mCE-inseparability is undecidable for \mathcal{SROIQ} [10], we present two sound but incomplete approximations to the problem of computing the minimal change set: ‘‘Subconcept’’ and ‘‘Grammar’’ diffs.

In addition, and in order to provide a basis for comparison between diff notions, we define the set of differences which would be captured by a comparison of the concept hierarchies between two ontologies, i.e. differences in atomic subsumptions, as $\text{AtDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$. Hereafter we refer to ContentCVS's diff notion as $\text{CvsDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$.

The first approximation, Subconcept diff (denoted $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$), is based on subconcepts of ontologies, wherein we check whether there is a difference in entailments of type $C \sqsubseteq D$, where C or D is a possibly complex concept from the set of Σ -subconcepts of \mathcal{O}_1 and \mathcal{O}_2 (see Definition 5). It is at least conceivable that many entailments will involve subconcepts, and, if that is the case, those would be witnesses that the user could understand, since they are explicitly asserted in either ontology. Moreover, this notion may exhibit entailment differences which would not show up if we restrict ourselves to either atomic subsumptions, or specific forms of entailments (in the manner of ContentCVS). The restriction to forms of concepts explicit in either ontology limits the amount of change captured. E.g., if we have $\mathcal{O}_1 = \{A \sqsubseteq \exists r.B\}$, and in \mathcal{O}_2 add an axiom $B \sqsubseteq \exists s.C$, the change $A \sqsubseteq \exists r.\exists s.C$ would not be found. However, the rationale behind this approach is that we could detect other kinds of change in a principled and relatively cheap way, e.g., $\mathcal{O}_1 = \{A \sqsubseteq B\}$, $\mathcal{O}_2 = \mathcal{O}_1 \cup \{B \sqsubseteq \exists r.(C \sqcap \exists r.D)\}$; we have that $\mathcal{O}_1 \not\models \alpha := A \sqsubseteq \exists r.(C \sqcap \exists r.D)$, while $\mathcal{O}_2 \models \alpha$.

In order to avoid only considering witnesses in their explicitly asserted form, we extend the previous diff notion and present Grammar diff (denoted $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$), which detects differences in additional types of entailments; the grammars below define the types of concepts taken into account by Grammar diff, where SC stands for a subconcept of $\mathcal{O}_1 \cup \mathcal{O}_2$.

Grammar G_L

$$C \longrightarrow SC \mid SC \sqcup SC \mid \exists r.SC \mid \forall r.SC \mid \neg SC$$

Grammar G_R

$$C \longrightarrow SC \mid SC \sqcap SC \mid \exists r.SC \mid \forall r.SC \mid \neg SC$$

The semantic difference between ontologies w.r.t. each mentioned diff notion is defined as follows:

Definition 5 *Given two ontologies and a signature Σ , the set of Σ -differences for a diff notion Φ is:*

$$\Phi\text{-Diff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma := \{\eta \in \Phi\text{-ax} \mid \mathcal{O}_1 \not\models \eta \wedge \mathcal{O}_2 \models \eta \wedge \tilde{\eta} \sqsubseteq \Sigma\}$$

where the set $\Phi\text{-ax}$ is defined as follows:

$$\begin{aligned} \text{if } \Phi = \text{At}, & \quad \{C \sqsubseteq D \mid C, D \in \Sigma\} \\ \text{if } \Phi = \text{Sub}, & \quad \{C \sqsubseteq D \mid C, D \text{ subconcepts in } \mathcal{O}_1 \cup \mathcal{O}_2\} \\ \text{if } \Phi = \text{Gr}, & \quad \{C \sqsubseteq D \mid D \text{ a concept over } G_L, \text{ or } C \text{ a concept over } G_R\} \\ \text{if } \Phi = \text{Cvs}, & \quad \{C \sqsubseteq D \mid C \in \Sigma \text{ and } D \text{ a concept over } G_{\text{cvs}}\} \\ \text{if } \Phi = \text{CEX}, & \quad \{C \sqsubseteq D \mid C, D \text{ subconcepts in } \mathcal{L}(\Sigma)\} \end{aligned}$$

It is not hard to see that there are subset relations between each diff and the actual $\text{MinCS}(\mathcal{O}_1, \mathcal{O}_2)$ that they approximate, as per Lemma 1:

Lemma 1 *Given two ontologies and a signature Σ :*

$$\begin{aligned} \text{AtDiff-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma &\subseteq \text{SubDiff-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \subseteq \text{GrDiff-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \subseteq \\ &\hspace{15em} \text{MinCS}(\mathcal{O}_1, \mathcal{O}_2) \\ \text{CvsDiff-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma &\subseteq \text{GrDiff-AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma \end{aligned}$$

The current implementation of CEX only takes as input acyclic \mathcal{ELH}^r terminologies, that is, \mathcal{ELH}^r TBoxes which are 1) acyclic and 2) every concept appears (alone) on the left-hand side of an axiom exactly once. In order to apply CEX to knowledge bases that are more expressive than \mathcal{ELH}^r terminologies, we rely on an approximation that uses CEX as a sub-routine.

Definition 6 (Approx-CEX) *Given two non- \mathcal{ELH}^r ontologies, the Approx-CEX procedure is:*

1. For $i \in \{1, 2\}$, approximate \mathcal{O}_i as an \mathcal{ELH}^r terminology, resulting in \mathcal{O}'_i :
 - (a) Remove all non- \mathcal{EL} axioms.
 - (b) Break cycles (non-deterministically).
 - (c) Remove all but one axiom with a given atomic left-hand side.
2. Apply CEX to $\mathcal{O}'_1, \mathcal{O}'_2$, resulting in a temporary change set: TempCS .
3. For each $\alpha \in \text{TempCS}$, add α to FinalCS if $\mathcal{O}_1 \not\models \alpha$ and $\mathcal{O}_2 \models \alpha$.
4. Return FinalCS ; the set of axioms in the diff.

Note that step 1 is parameterizable with any \mathcal{ELH}^r approximation algorithm. Additionally, step 2 can be replaced with a diff implementation for more expressive logics, with either the input approximation (step 1) and soundness check (step 3) removed, or with an altered step 1 depending on the expressivity of the input. Step 4 in Definition 6 is necessary to ensure that changes detected within the \mathcal{ELH}^r approximations (obtained in step 1) are sound changes w.r.t. the whole ontologies. Obviously, this approximation-based procedure throws away a lot of information and is not deterministic. However, even such an approximation can offer useful insight, esp. if it finds changes that other methods do not. There are more elaborate existing approximation approaches (e.g., [13]), but they generally do not produce \mathcal{ELH}^r terminology, so their use requires either changing the approximation output or updating CEX to take non-terminological \mathcal{EL} input.

5 Empirical Results

The object of our evaluation is a subset of the NCIt corpus used in [4], with expressivity ranging from $\mathcal{ALCH}(\mathcal{D})$ to $\mathcal{SH}(\mathcal{D})$. More specifically, we take into account 12 versions of the NCIt which contain concept-based change logs. In order to investigate the applicability of our approach we (1) compare the results obtained via our approximations with those output by Approx-CEX and ContentCVS, and (2) inspect whether the devised approximations capture any direct changes not reported in the NCIt change logs.

The experiment machine used is an Intel Xeon Quad-Core 3.20GHz, with 16Gb DDR3 RAM. The system runs Mac OS X 10.6.8, Java Virtual Machine (JVM v1.5), and all tests were run using the OWL API (v3.2.4) [5].³

In terms of computation times, on average computing $\text{AtDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ takes ≈ 20 seconds, Approx-CEX takes ≈ 9 minutes, while computing $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ takes ≈ 35 minutes. The computation of $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ takes ≈ 14 hours for a subset of the ontology signature of size ≈ 1800 concept names, and ContentCVS ≈ 10 hours on the same randomly selected signature as $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$.⁴

5.1 Diff Comparison

The comparison of each diff w.r.t. number of affected concept names found is shown in Table 1, which displays the number of specialised concepts (L-changes), generalised concepts (R-changes), and the total number of affected concepts. Figure 1 shows a comparison of the number of affected concept names found by ContentCVS and Grammar diff within the randomly selected signatures. Note that, at this point, no distinction is made between direct and indirect changes.

Table 1: Number of affected concept names found by each diff, and their respective coverage w.r.t. affected concepts found by GrammarDiff.

NCIt	Approx-CEX			AtDiff			SubconceptDiff			GrammarDiff		
	L	R	Total	L	R	Total	L	R	Total	L	R	Total
1 (05.07d)	454	307	668	979	486	1,416	1,701	490	2,131	10,501	3,597	12,178
2 (05.10e)	413	648	851	792	499	1,208	1,436	518	1,816	11,366	3,442	12,975
3 (05.11f)	3,508	2,089	5,013	5,233	1,172	6,135	5,910	1,178	6,528	12,379	6,806	17,542
4 (05.12f)	1,400	2,813	2,950	2,358	1,485	3,676	45,825	1,495	45,932	19,547	13,691	28,305
5 (06.01c)	7,305	2,495	8,692	3,808	1,321	4,978	15,254	1,498	15,691	36,333	20,137	39,491
6 (06.02d)	1,131	684	1,520	3,502	624	3,923	5,806	663	6,203	10,621	11,331	19,741
7 (06.03d)	1,721	2,434	3,052	2,462	1,127	3,217	5,777	1,201	6,330	20,620	9,799	24,567
8 (06.04d)	417	1,382	1,590	6,284	1,631	6,806	6,952	1,674	7,428	10,275	7,576	14,047
9 (06.05d)	1,095	1,455	1,711	2,224	678	2,745	4,928	737	5,329	13,291	9,223	13,819
10 (06.06e)	1,649	1,002	2,154	4,073	607	4,553	5,992	663	6,415	26,161	5,345	28,005
11 (06.08d)	624	968	1,099	1,240	610	1,714	3,910	731	4,325	37,674	3,630	38,502
Avg. Cov.	9%	18%	12%	20%	12%	18%	52%	13%	41%			
Min. Cov.	2%	6%	3%	3%	6%	4%	10%	6%	11%			
Max. Cov.	28%	31%	29%	61%	22%	48%	100%	22%	100%			

Due to computational issues regarding Grammar diff and ContentCVS , instead of comparing each pair of NCIt versions w.r.t. $\Sigma = \tilde{\mathcal{O}}_1 \cup \tilde{\mathcal{O}}_2$ we take a random sample of the terms in the ontology (generally $n \approx 1800$) such that a straightforward extrapolation allows us to determine that the true proportion of changed terms lies in the confidence interval ($\pm 3\%$) with a 99% confidence level. In general, Grammar diff, even taking into account the confidence interval, consistently detects more changes (both L and R) than all other diffs. Also,

³ <http://owlapi.sourceforge.net/>

⁴ Note that, originally, ContentCVS only computes $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma^L$, but in order to provide a direct comparison with the diffs here proposed we also compute $\text{AT}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma^R$ according to ContentCVS 's grammar.

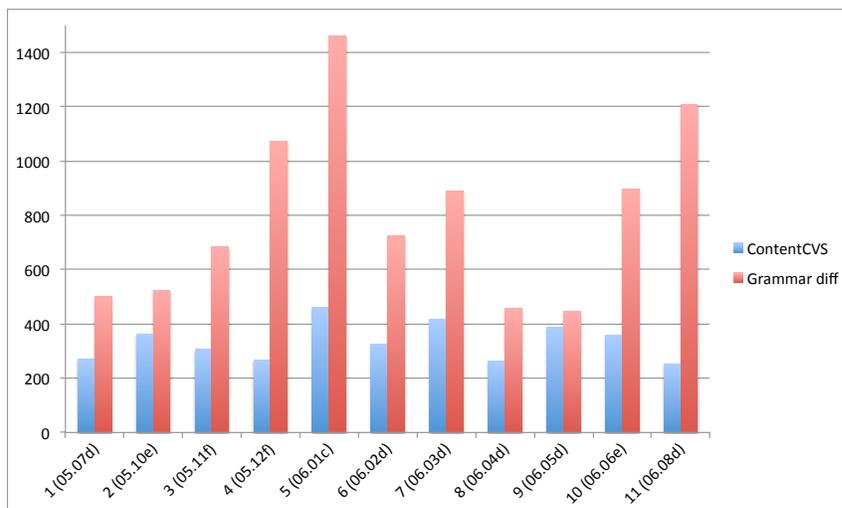


Fig. 1: Comparison of total number of affected concepts found by ContentCVS and Grammar diff (y -axis: number of concept names, x -axis: NCIIt version).

despite the one case where the lower bound of detected changes is inferior to another diff, in version 4, it cannot be worse than Subconcept diff by Lemma 1.

5.2 Direct Changes in the NCIIt Logs

The change logs supplied with each version of the NCIIt contain those concept names which were subject to changes. However, it is unclear whether each reported change also (or solely) relates to annotation changes. It could be the case that a reported concept change is purely ineffectual. In spite of this ambiguity, it should be expected that a change log contains concept names that were directly changed, and this is what we aim to find out in our next experiment; we extract the concept names mentioned in the change log, and verify whether the obtained direct changes for each NCIIt version are contained in said change logs. The results are shown in Table 2, where the affected concept names shown in Section 5.1 are partitioned into purely direct, purely indirect, or both directly and indirectly changed concepts. Overall, we see that the change logs do miss a lot of direct changes, more specifically, on average, $\text{AtDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ reveals 767 changed concept names not mentioned in the change logs, while $\text{SubDiff}(\mathcal{O}_1, \mathcal{O}_2)_\Sigma$ uncovers 908 such concept names per NCIIt version.

6 Discussion

First thing to notice is that SubDiff finds many more changes than AtDiff and Approx-CEX, while often not reaching close to the projected values of GrammarDiff (the average coverage being 41%). The latter, as expected, captures far more changes within the selected signatures than ContentCVS.

Table 2: Number of purely direct (P.D.), purely indirect (P.I.), and both directly and indirectly (Mix) changed concepts. Number of directly changed concepts that do not appear in the NCIt change logs (denoted Missed).

NCIt	AtDiff							SubDiff						
	L			R			Missed	L			R			Missed
	Mix	P.D.	P.I.	Mix	P.D.	P.I.		Mix	P.D.	P.I.	Mix	P.D.	P.I.	
1	524	122	333	88	206	192	798	686	134	881	88	210	192	953
2	440	125	227	95	179	225	149	803	344	289	96	198	224	211
3	2,106	215	2,912	211	680	281	315	2,242	549	3,119	212	686	280	445
4	1,498	126	734	146	1,041	298	190	2,647	78	43,100	148	1,050	297	432
5	1,401	154	2,253	127	882	312	243	6,511	1,527	7,216	304	882	312	317
6	813	77	2,612	153	232	239	199	1,163	143	4,500	161	240	262	199
7	984	206	1,272	256	448	423	273	2,400	320	3,057	267	513	421	511
8	5,923	152	209	154	1,267	210	5,546	5930	483	539	157	1,308	209	5,723
9	870	611	743	166	254	258	207	1,775	832	2,321	171	307	259	322
10	594	2,727	752	145	225	237	216	2,110	2,854	1,028	147	280	236	298
11	586	167	487	139	239	232	300	1,050	354	2,506	147	325	259	582

Considering the high number of affected concepts found by SubDiff in versions 4 and 5 of the NCIt, one can argue that analysing such a change set would be rather unpleasant. By categorising concept names in the change set according to whether they are directly or indirectly affected, we can greatly reduce the information overload; notice that, e.g., in version 4 there are 45,825 specialised concepts, out of which there are only 78 purely directly changed concepts, and the majority of the remainder are purely indirect changes (43,100). Similarly in version 5, from 15,254 specialised concepts there are only 1,527 purely direct changes. Immediately we see that this mechanism can provide an especially helpful means to assist change analysis, by, e.g., confining the changes shown upfront to only those which are (purely) direct.

Despite the optimisations applied in GrammarDiff’s implementation, e.g., for $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^L$ we start by verifying whether there exists some effectual change between $\perp\text{-mod}(\{A\}, \mathcal{O}_1)$ and $\perp\text{-mod}(\{A\}, \mathcal{O}_2)$, for each $A \in \Sigma$ (analogously we use $\top\text{-modules}$ for $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}^R$), only considering witnesses whose signature is contained in the module signature, stopping once we find a single witness for a concept name, the computation of $\text{GrDiff}(\mathcal{O}_1, \mathcal{O}_2)_{\Sigma}$ still takes long, and needs further optimisations. The major bottleneck is that the $\top\text{-modules}$ for a concept name provide too big an approximation, e.g., for a top-level concept its $\top\text{-module}$ contains almost the whole ontology. Thus $\top\text{-modules}$ do not restrict much of our search space, not at least in the same way as $\perp\text{-modules}$ do. Additionally, in order to take advantage of the categorisation mechanism proposed, we would need to compute all witnesses for each $\Sigma\text{-concept}$ (which is relatively cheap in SubDiff).

7 Conclusions

We have formulated the problem of finding the set of affected terms between ontologies via model inseparability, and presented feasible approximations to finding this set. We have shown that each of the approximations can find considerably

more changes than those visible in a comparison of concept hierarchies. Both sound approximations devised capture more changes than Approx-CEX. The restrictions imposed by CEX on the input ontologies make change-preserving approximations a challenge, as we have seen in our attempt to reduce the NCIt to \mathcal{EL} in a less naive way.

The proposed distinction between (purely) direct and indirect allows users to focus on those changes which are specific to a given concept, in addition to masking possibly uninteresting changes to any and all concept names (such as those obtained via witnesses constructed with negation and disjunction), thereby making change analysis more straightforward. As demonstrated by the NCIt change log analysis, we have found a (often high) number of direct changes that are not contained in the NCIt change logs, which leads us to believe the recording of changes does not seem to follow from even a basic concept hierarchy comparison, but rather a seemingly ad hoc mechanism.

In future work we aim to optimise the devised approximations so as to compare all NCIt versions w.r.t. their signature union, and deploy an end-user tool.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. *J. of Artificial Intelligence Research* 31 (2008)
3. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Proc. of KR-06 (2006)
4. Gonçalves, R.S., Parsia, B., Sattler, U.: Categorising logical differences between OWL ontologies. In: Proc. of CIKM-11 (2011)
5. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: Proc. of OWLED-09 (2009)
6. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga Llavori, R.: Supporting concurrent ontology development: Framework, algorithms and tool. *Data and Knowledge Engineering* 70(1) (2011)
7. Konev, B., Walther, D., Wolter, F.: The logical difference problem for description logic terminologies. In: IJCAR-08. vol. 5195 (2008)
8. Křemen, P., Šmíd, M., Kouba, Z.: OWLDiff: A practical tool for comparison and merge of OWL ontologies. In: Proc. of DEXA-12 (2011)
9. Küsters, R.: Non-Standard Inferences in Description Logics, LNAI, vol. 2100. Springer-Verlag (2001)
10. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of IJCAI-07 (2007)
11. Lutz, C., Wolter, F.: Conservative extensions in the lightweight description logic \mathcal{EL} . In: Proc. of CADE-21 (2007)
12. Malone, J., Holloway, E., Adamusiak, T., Kapushesky, M., Zheng, J., Kolesnikov, N., Zhukova, A., Brazma, A., Parkinson, H.E.: Modeling sample variables with an experimental factor ontology. *Bioinformatics* 26(8), 1112–1118 (2010)
13. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness Preserving Approximation for TBox Reasoning. In: Proc. of AAAI-10 (2010)
14. Sattler, U., Schneider, T., Zakharyashev, M.: Which kind of module should I extract? In: Proc. of DL-09 (2009)

Equality-Friendly Well-Founded Semantics and Applications to Description Logics

Georg Gottlob^{1,2}, André Hernich³, Clemens Kupke¹, and Thomas Lukasiewicz¹

¹ Department of Computer Science, University of Oxford, UK
{firstname.lastname}@cs.ox.ac.uk

² Oxford-Man Institute of Quantitative Finance, University of Oxford, UK

³ Institut für Informatik, Humboldt-Universität zu Berlin, Germany
hernich@informatik.hu-berlin.de

Abstract. We tackle the problem of defining a well-founded semantics (WFS) for Datalog rules with existentially quantified variables in their heads and negations in their bodies. In particular, we provide a WFS for the recent Datalog[±] family of ontology languages, which covers several important description logics (DLs). To do so, we generalize Datalog[±] by non-stratified nonmonotonic negation in rule bodies, and we define a WFS for this generalization via guarded fixed point logic. We refer to this approach as *equality-friendly WFS*, since it has the advantage that it does not make the unique name assumption (UNA); this brings it close to OWL and its profiles as well as typical DLs, which also do not make the UNA. We prove that for guarded Datalog[±] with negation under the equality-friendly WFS, conjunctive query answering is decidable, and we provide precise complexity results for this problem. From these results, we obtain precise definitions of the standard WFS extensions of \mathcal{EL} and of members of the *DL-Lite* family, as well as corresponding complexity results for query answering.

1 Introduction

The recent Datalog[±] family of ontology languages [7] extends plain Datalog by the possibility of existential quantification in rule heads and other features, and simultaneously restricts the rule syntax to achieve tractability. The following example illustrates how description logic (DL) knowledge bases are expressed in Datalog[±].

Example 1 (Literature) The knowledge that every conference paper is an article and that every scientist is the author of at least one paper can be expressed in DL by the TBox axioms $ConferencePaper \sqsubseteq Article$ and $Scientist \sqsubseteq \exists isAuthorOf$, respectively, while the knowledge that John is a scientist can be expressed by the ABox axiom $Scientist(john)$. In Datalog[±], the former are encoded as the rule $ConferencePaper(X) \rightarrow Article(X)$ and the rule $Scientist(X) \rightarrow \exists Y isAuthorOf(X, Y)$, respectively, and the latter is encoded by an identical fact in the database. Furthermore, the TBox axiom that encodes that conference papers are not journal papers, can be expressed in Datalog[±] by the negative constraint $ConferencePaper \wedge JournalPaper \rightarrow \perp$. A simple Boolean conjunctive query (BCQ) asking whether John authors a paper is $\exists X isAuthorOf(john, X)$. ■

The Datalog[±] languages bridge an apparent gap in expressive power between database query languages and DLs as ontology languages, extending the well-known Dat-

alog language in order to embed DLs. They also allow for transferring important concepts and proof techniques from database theory to DLs. For example, it was so far not clear how to enrich tractable DLs by the feature of nonmonotonic negation. By the results of [7], DLs can be enriched by stratified negation via mappings from DLs to Datalog^\pm with stratified negation, which is defined and studied in that paper. Given that stratified negation is quite limited, we wondered whether the richer and more expressive well-founded negation could be defined for Datalog^\pm . The well-founded semantics (WFS) for normal (logic) programs [25] is one of the most widely used semantics for nonmonotonic normal programs, it is the standard semantics for such programs for database applications, and it is thus especially under a data-oriented perspective of great importance for the Web. Having many nice features, the WFS is defined for all normal programs (i.e., logic programs with the possibility of negation in rule bodies), has a polynomial data tractability, approximates the answer set semantics, and coincides with the canonical model in case of stratified normal programs.

In this paper, we concentrate on the important problem of defining a WFS for (unrestricted) normal Datalog^\pm , i.e., Datalog with existentially quantified variables in rule heads and negations in rule bodies. This new semantics is called the *equality-friendly WFS (EFWFS)*, since it has the crucial advantage that it does not make the unique name assumption (UNA); this brings it close to OWL and its profiles as well as typical DLs, which also do not make the UNA.

Since (unrestricted) normal Datalog^\pm generalizes positive Datalog^\pm , consistency checking and query answering in it is in general undecidable. However, it turns out that the guarded fragment of normal Datalog^\pm can be translated to *guarded fixed point logic*, which is a well-studied decidable formalism. Through this translation, we thus obtain the decidability of consistency checking and query answering in guarded normal Datalog^\pm . Furthermore, we obtain upper complexity bounds, which are then also shown to be tight. Guarded Datalog^\pm covers in particular the DLs \mathcal{EL} and $DL\text{-Lite}_{\mathcal{R}}$ (which is underlying the OWL 2 QL profile). Therefore, our decidability results and upper complexity bounds carry over to these DLs. The following example illustrates how the WFS can be extended to such DLs.

Example 2 (Holidays) Consider an ABox containing the three holiday destinations $Dest(d_1)$, $Dest(d_2)$, and $Dest(d_3)$. Suppose that any destination that offers the opportunity to swim needs either direct access to the beach ($Beach(x)$) or a bus connection to some beach ($BeachBus(x,y)$). That is, at destinations where swimming is possible, we want to make sure that never both $\text{not}Beach$ and $\text{not}\exists BeachBus$ hold. This can be achieved by the following two rules:

$$Dest \sqcap Swimming \sqcap \text{not}Beach \sqsubseteq \exists BeachBus; \quad (1)$$

$$Dest \sqcap Swimming \sqcap \text{not}\exists BeachBus \sqsubseteq Beach. \quad (2)$$

Observe also that $\text{not}Swimming(d)$ would immediately imply the facts $\text{not}Beach(d)$ and $\text{not}\exists BeachBus(d)$, since it would make it impossible that either of the two axioms could be applied to derive new facts about d . ■

The following example shows a case where not making the UNA is more appropriate than making it under the WFS.

Example 3 (Company) Suppose we are given certain facts about employees and their employers. The following two concept membership axioms state that John and Sam are employees: $Employee(John)$, $Employee(Sam)$. To these axioms, we add a concept inclusion axiom that maintains that every employee must have an employer: $Employee \sqsubseteq \exists.hasEmployer$. Finally, we would like to test whether or not *John* and *Sam* work in the same company, which is expressed by the query $\exists x(hasEmployer(John,x) \sqcap notHasEmployer(Sam,x))$. Then, under the UNA, equality between all individuals (including new ones) is minimized, and we evaluate the query to true, which is not the case without UNA, where different Skolem terms may be interpreted by the same object. ■

2 Preliminaries

In this section, we briefly recall some basics on Datalog[±] [7].

Databases and Queries. We assume (i) an infinite universe of (*data*) constants Δ (which constitute the “normal” domain of a database), and (ii) an infinite set of variables \mathcal{V} (used in queries and constraints). We denote by \mathbf{X} sequences of variables X_1, \dots, X_k with $k \geq 0$. We assume a *relational schema* \mathcal{R} , which is a finite set of *relation names* (or *predicate symbols*, or simply *predicates*). A *term* t is a constant or variable. An *atomic formula* (or *atom*) \mathbf{a} has the form $P(t_1, \dots, t_n)$, where P is an n -ary predicate, and t_1, \dots, t_n are terms. A conjunction of atoms is often identified with the set of all its atoms.

A *database (instance)* D for a relational schema \mathcal{R} is a (possibly infinite) set of atoms with predicates from \mathcal{R} and arguments from Δ . A *conjunctive query (CQ)* over \mathcal{R} has the form $Q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms with the variables \mathbf{X} and \mathbf{Y} , and eventually constants. Note that $\Phi(\mathbf{X}, \mathbf{Y})$ may also contain equalities but no inequalities. A *Boolean CQ (BCQ)* over \mathcal{R} is a CQ of the form $Q()$. We often write a BCQ as the set of all its atoms, having constants and variables as arguments, and omitting the quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu: \Delta \cup \mathcal{V} \rightarrow \Delta \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, and (ii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$ over a database D , denoted $Q(D)$, is the set of all tuples \mathbf{t} over Δ for which there exists a homomorphism $\mu: \mathbf{X} \cup \mathbf{Y} \rightarrow \Delta$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Tuple-Generating Dependencies (TGDs). Tuple-generating dependencies (TGDs) describe constraints on databases in the form of generalized Datalog rules with existentially quantified conjunctions of atoms in rule heads; their syntax and semantics are as follows. Given a relational schema \mathcal{R} , a *tuple-generating dependency (TGD)* σ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over \mathcal{R} , called the *body* and the *head* of σ , respectively. Such σ is satisfied in a database D for \mathcal{R} iff, whenever there is a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D , there is an extension h' of h that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D . Since TGDs can be reduced to TGDs with only single atoms in their heads, in the sequel, every TGD has w.l.o.g. a single atom in its head. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ . The leftmost such atom is the *guard* of σ .

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for \mathcal{R} , and a set of TGDs Σ on \mathcal{R} , the set of *models* of D and Σ , denoted $\text{mods}(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B . The set of *answers* for a CQ Q to D and Σ , denoted $\text{ans}(Q, D, \Sigma)$, is the set of all tuples \mathbf{a} such that $\mathbf{a} \in Q(B)$ for all $B \in \text{mods}(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $\text{ans}(Q, D, \Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [5], even with fixed schema and TGDs [6].

Negative Constraints. Another crucial ingredient of Datalog^\pm for ontological modeling are *negative constraints* (or simply *constraints*), which are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \perp$, where $\Phi(\mathbf{X})$ is a conjunction of atoms (not necessarily guarded), called its *body*. We usually omit the universal quantifiers, and we implicitly assume that all sets of constraints are finite here.

Normal TGDs and BCQs. Normal TGDs are TGDs that may also contain (default-) negated atoms in their bodies: Given a relational schema \mathcal{R} , a *normal TGD (NTGD)* σ has the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms and negated atoms over \mathcal{R} , and $\Psi(\mathbf{X}, \mathbf{Z})$ is a conjunction of atoms over \mathcal{R} . We usually omit the universal quantifiers. As for standard TGDs, w.l.o.g., $\Psi(\mathbf{X}, \mathbf{Z})$ is a singleton atom. We say σ is *satisfied* in a database D for \mathcal{R} iff, whenever there is a homomorphism h for all the variables and constants in the body of σ that maps (i) positive literals of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D and (ii) no negated atom of $\Phi(\mathbf{X}, \mathbf{Y})$ to an atom of D , then there is an extension h' of h that maps the head atom to an atom of D . We call σ *guarded* iff it contains a positive body atom that contains all universally quantified variables of σ . W.l.o.g., constants in the body of guarded σ occur only in guards.

We next add negation to BCQs. A *normal Boolean conjunctive query (NBCQ)* Q is an existentially closed conjunction of atoms and negated atoms $\exists \mathbf{X} p_1(\mathbf{X}) \wedge \dots \wedge p_m(\mathbf{X}) \wedge \neg p_{m+1}(\mathbf{X}) \wedge \dots \wedge \neg p_{m+n}(\mathbf{X})$, where $m \geq 1$, $n \geq 0$, and the variables of the p_i 's are among \mathbf{X} . Q is *covered* if for every negative atom α in Q there is a positive atom in Q containing every argument in α . In the sequel, w.l.o.g., NBCQs contain no constants.

3 Equality-Friendly WFS for Datalog^\pm

In this section, we first recall the well-founded semantics (WFS) of normal programs. As a central new contribution, we then introduce a WFS of normal Datalog^\pm programs without the UNA.

WFS of Normal Programs. The WFS [25] is the most widely used semantics for non-monotonic programs, it is the standard semantics for such programs for database applications, and it is thus especially under a data-oriented perspective of great importance for the Web. For our purposes, it is enough to recall the WFS of *function-free ground* normal programs, and we refer to [25] for the general case.

We first give some preliminary definitions. A function-free normal program is a finite set of rules r of the form $\beta_1, \dots, \beta_n, \neg \beta_{n+1}, \dots, \neg \beta_{n+m} \rightarrow \alpha$, where $\alpha, \beta_1, \dots, \beta_{n+m}$ are atoms and $m, n \geq 0$. We call α the *head* of r , denoted $H(r)$, while the conjunction $\beta_1, \dots, \beta_n, \neg \beta_{n+1}, \dots, \neg \beta_{n+m}$ constitutes its *body*. Let $B(r) = B^+(r) \cup B^-(r)$, where

$B^+(r) = \{\beta_1, \dots, \beta_n\}$, and $B^-(r) = \{\beta_{n+1}, \dots, \beta_{n+m}\}$. The program is *ground* if it contains no variables. Let P be a function-free ground normal program. The *Herbrand base* of P , denoted HB_P , is the set of all atoms that can be constructed from the predicate symbols and the constants appearing in P . For all $S \subseteq HB_P$, we let $\neg.S = \{\neg.a \mid a \in S\}$. We denote by $Lit_P = HB_P \cup \neg.HB_P$ the set of all literals with predicate symbols and constants from P . A (*three-valued*) *interpretation* relative to P is any set $I \subseteq Lit_P$ that is *consistent* (i.e., there is no atom $a \in HB_P$ with $\{a, \neg.a\} \subseteq I$).

The WFS has many different equivalent definitions (see also [4]). We here recall the one based on unfounded sets, via the operators U_P , T_P , and W_P . A set $U \subseteq HB_P$ is an *unfounded set* of P relative to $I \subseteq Lit_P$ iff for every $a \in U$ and every rule $r \in P$ with $H(r) = a$, either (i) $\neg.b \in I \cup \neg.U$ for some atom $b \in B^+(r)$, or (ii) $b \in I$ for some atom $b \in B^-(r)$. There exists the greatest unfounded set of P relative to I , denoted $U_P(I)$. Intuitively, it collects all those atoms that cannot become true when extending I with further information. We are now ready to define the two operators T_P and W_P on interpretations $I \subseteq Lit_P$ relative to P by:

$$T_P(I) = \{H(r) \mid r \in P, B^+(r) \cup \neg.B^-(r) \subseteq I\}; \quad W_P(I) = T_P(I) \cup \neg.U_P(I).$$

Since W_P is monotonic, it has a least fixed point, denoted $lfp(W_P)$, which is the *well-founded semantics* (WFS) of P , denoted $WFS(P)$. Intuitively, starting with $I = \emptyset$, rules are applied to obtain new positive and negated facts (via $T_P(I)$ resp. $\neg.U_P(I)$). This is repeated until no longer possible.

EFWFS of Normal Datalog[±] Programs. We relate normal Datalog[±] programs to sets of function-free ground normal programs, and define their equality-friendly WFS (EFWFS) as the set of well-founded models of the associated normal programs.

The basic idea is as follows. If we do not make the UNA, different constants in a normal Datalog[±] program P may represent the same value. Thus, P may turn out to be any of the programs P' obtained from P by identifying constants. Furthermore, in every such program P' , existential quantifiers may introduce one or more value, which, since we do not make the UNA, does not have to be “fresh”, but can be any constant. Hence, without the UNA, the meaning of P may be captured by the set of all normal programs P'' obtained from P by identifying values, and replacing TGDs in P by arbitrary instances, at least one for each possible variable assignment for its body. It is then natural to consider the well-founded models of all those programs P'' as the semantics of P .

More precisely, an *instance* of a normal TGD $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$ is a rule of the form $\Phi(\mathbf{a}, \mathbf{b}) \rightarrow \Psi(\mathbf{a}, \mathbf{c})$, where $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are tuples of constants. Let $P = D \cup \Sigma$ be a normal Datalog[±] program, where D is a database and Σ a set of normal TGDs. An *instance* \mathcal{I} of P is a normal program consisting of all facts in D , and instances of TGDs in Σ such that for all TGDs $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$ in Σ , and all interpretations \mathbf{a}, \mathbf{b} for \mathbf{X}, \mathbf{Y} , there is at least one \mathbf{c} such that $\Phi(\mathbf{a}, \mathbf{b}) \rightarrow \Psi(\mathbf{a}, \mathbf{c})$ is in \mathcal{I} . Let $\mathcal{I}(P)$ be the set of all instances of P , and $dom(P)$ the set of all constants in P . For any $\mu: dom(P) \rightarrow \Delta$, let P_μ be the program obtained from P by replacing all constants c with $\mu(c)$. Then, the *equality-friendly well-founded semantics* of P , denoted $EFWFS(P)$, is the set $\{WFS(P'') \mid \mu: dom(P) \rightarrow \Delta, P'' \in \mathcal{I}(P_\mu)\}$. Query-answering for an NBCQ Q is now defined by saying that Q evaluates to *Yes* in $EFWFS(P)$ iff Q evaluates to *Yes* in all elements of $EFWFS(P)$. Here, an NBCQ Q evaluates to *Yes* in a

three-valued interpretation I iff there is a homomorphism that maps all elements of Q^+ into atoms in I and all elements of Q^- into negated atoms in I .

Example 4 Consider the following program P :

$$\begin{array}{l} \rightarrow A(0); \quad A(X) \rightarrow \exists Y_1, Y_2 (R(X, Y_1, Y_2)); \\ R(X_1, X_2, X_3), \neg A(X_3) \rightarrow S(0); \quad A(X), \neg S(X) \rightarrow \exists Y_1, Y_2 (R(Y_1, Y_2, X)). \end{array}$$

Obviously, $A(0)$ is in the EFWFS of P . Furthermore, because of the second rule, every possible well-founded model of P contains $R(0, c_1, c_2)$. But we cannot assume $c_2 \neq 0$. If $c_2 = 0$, all possible applications of the third rule are blocked, and thus $\neg S(0)$ would be derived (the last rule may then still add another atom $R(d_1, d_2, 0)$, but this does not affect the overall result). Otherwise, in case $c_2 \neq 0$, the third rule would yield $S(0)$, because clearly we have $\neg A(c_2)$. Therefore, neither $S(0)$ nor its negation is in $EFWFS(P)$ and the only atoms that are always in $EFWFS(P)$ are $A(0)$ and $R(0, c_1, c_2)$ for some constants c_1 and c_2 (so, the query $\exists X_1, X_2 (R(0, X_1, X_2))$ would evaluate to true). The picture changes, if we add the constraint $R(X_1, X_2, X_3), R(X_3, X_2, X_1) \rightarrow \perp$. Then, c_2 generated by the second rule must be different from 0, and so we always obtain $S(0)$, and we get $\neg R(d_1, d_2, 0)$, for all d_1, d_2 , by the last rule. ■

4 Translation into Guarded Fixed Point Logic

The EFWFS can be characterized in terms of *guarded fixed point logic*. This characterization turns out to be more convenient for reasoning about the EFWFS of guarded normal Datalog[±] programs.

Guarded fixed point logic (GFP), introduced by Grädel and Walukiewicz [13], simultaneously restricts and extends first-order logic by enforcing a certain quantification pattern, and allowing for inductively defining relations, while having a satisfiability problem of moderate complexity.

Let \mathcal{R} be a relational schema. The set of formulas of GFP over \mathcal{R} is built from atomic formulas over \mathcal{R} (including equality atoms) using Boolean combinations, and the following two additional formula formation rules:

- I. If α is an atomic formula over \mathcal{R} containing the variables in \mathbf{X} , and ψ is a GFP formula over \mathcal{R} whose free variables occur in α , then $\exists \mathbf{X} (\alpha \wedge \psi)$ and $\forall \mathbf{X} (\alpha \rightarrow \psi)$ are GFP formulas over \mathcal{R} . The formula α is called *guard*.
- II. Let R be a k -ary predicate, \mathbf{X} a k -tuple of variables, and $\psi(R, \mathbf{X})$ a GFP formula over $\mathcal{R} \cup \{R\}$ whose free variables occur in \mathbf{X} , and where R appears only positively (in the scope of an even number of negation symbols) and not in guards. Then, $[\mathbf{lfp}_{R, \mathbf{X}} \psi](\mathbf{X})$ and $[\mathbf{gfp}_{R, \mathbf{X}} \psi](\mathbf{X})$ are GFP formulas over \mathcal{R} with free variables \mathbf{X} .

As for the semantics of the formulas in II, given a database D for \mathcal{R} , ψ defines an operator $F: \mathcal{P}(\text{dom}(D)^k) \rightarrow \mathcal{P}(\text{dom}(D)^k)$ with $F(S) := \{\mathbf{a} \mid D \models \psi(S, \mathbf{a})\}$. This operator is monotone and thus has a least fixed point $\mathit{lfp}(F)$ and a greatest fixed point $\mathit{gfp}(F)$. Then, $D \models [\mathbf{lfp}_{R, \mathbf{X}} \psi](\mathbf{a})$ iff $\mathbf{a} \in \mathit{lfp}(F)$, and $D \models [\mathbf{gfp}_{R, \mathbf{X}} \psi](\mathbf{a})$ iff $\mathbf{a} \in \mathit{gfp}(F)$.

Example 5 ([13]) The following GFP sentence says that the binary relation E is well-founded (i.e., no element is the endpoint of an infinite E -path): $\forall x, y (E(x, y) \rightarrow [\mathbf{lfp}_{W, x} \forall y (E(y, x) \rightarrow W(y))](x))$. ■

We are now ready to describe the translation of guarded normal Datalog[±] into GFP. Let $P = D \cup \Sigma$ be a fixed guarded normal Datalog[±] program, where D is a database, and Σ is a set of guarded NTGDs. Without loss of generality, we assume that P contains only a single predicate R ;⁴ let k be its arity. We construct a GFP sentence $\text{efwfs}(P)$ whose models closely correspond to the databases in $\text{EFWFS}(P)$. The key is to “existentially quantify” all the instances of NTGDs that we use to compute the WFS, and to mimic the fixed-point definition of the WFS using those instances.

Technically, we represent instances of an NTGD $\sigma \in \Sigma$ using a $2k$ -ary predicate S_σ . If $R(\mathbf{U})$ is the guard of σ , and $R(\mathbf{V})$ is its head, then a tuple (\mathbf{a}, \mathbf{b}) in S_σ encodes the instance of σ obtained by substituting \mathbf{a} and \mathbf{b} for \mathbf{U} and \mathbf{V} , respectively. For example, if σ is the NTGD $R(X, Y, 0), \neg R(X, 1, 1) \rightarrow \exists Z R(Z, X, 2)$, then the atom $S_\sigma((a, b, 0), (c, a, 2))$ represents the instance $R(a, b, 0), \neg R(a, 1, 1) \rightarrow R(c, a, 2)$ of σ .

We also use k -ary predicates T^*, C, T, F , and R , where T^* is intended to encode a superset of all true atoms; C holds all permutations of tuples in T^* ; T and F respectively hold the set of all true atoms and the set of all false atoms (the latter relativized to C); and R corresponds to the predicate R of the initial database D . It is not hard to construct GFP sentences that enforce the desired interpretation of the predicates S_σ, T^*, C , and R . Based on those GFP sentences, it is then possible to construct GFP sentences ψ_T and ψ_F that enforce the desired interpretations of T and F . The sentence ψ_T basically does nothing else than defining the (set of all positive literals of the) least fixed point of $W_{P'}$, where P' consists of all instances of NTGDs in Σ represented by the tuples in the predicates S_σ , while ψ_F defines the greatest fixed point of a certain operator, yielding the greatest unfounded set $U_{P'}(T)$ (relativized to C).

For an NBCQ Q over $\{R\}$, let Q^* be the BCQ obtained by replacing every positive literal $R(\mathbf{X})$ in Q with $T(\mathbf{X})$, and every negative literal $\neg R(\mathbf{X})$ in Q with $F(\mathbf{X})$.

Lemma 6 *For all covered NBCQs Q over the schema of P , we have $\text{EFWFS}(P) \models Q$ iff $\text{efwfs}(P) \models Q^*$.*

5 Complexity

We now take a look at the complexity of answering covered NBCQs over guarded normal Datalog[±] programs $P = D \cup \Sigma$. More precisely, we consider the *combined complexity*, measured in terms of the overall input, including P and the query.

In Section 4, we characterized the EFWFS of guarded normal Datalog[±] programs via a translation into guarded fixed point logic GFP. From the fact that satisfiability for GFP sentences is in 2-EXPTIME (and in EXPTIME in case of bounded arities) [13], we obtain the following:

Theorem 7 *Deciding $\text{EFWFS}(P) \models Q$, where $P = D \cup \Sigma$ is a guarded normal Datalog[±] program and Q a covered NBCQ, is 2-EXPTIME-complete in the general case, and EXPTIME-complete in the case of bounded arities and acyclic Q . Hardness holds even with respect to atomic queries.*

⁴ It is an easy task to transform a guarded normal Datalog[±] program into this form, since multiple predicates can be simulated by constants and a single predicate. For example, $A(x, y) \wedge B(x)$ may be replaced by $R(a, x, y) \wedge R(b, x, x)$, where a and b are extra constant symbols.

We remark that Theorem 7 carries over to *unions of covered NBCQs*, that is, queries Q of the form $\bigvee_{i=1}^n Q_i$, where each Q_i is a covered NBCQ.

6 WFS for OWL 2 QL

All the DLs of the *DL-Lite* family in [8,22] and the DL \mathcal{EL} [3] can be embedded into Datalog[±] [7]. In particular, this holds for *DL-Lite_R*, which forms the theoretical basis of the QL profile of the Web ontology language OWL 2. Both our equality-friendly WFS (EFWFS) for normal Datalog[±] and the OWL 2 QL profile do not make the UNA. Our work in this paper thus paves the way for an extension of OWL 2 QL with nonmonotonic negation under the EFWFS.

The following definition extends *DL-Lite_R*⁵ (underlying the OWL 2 QL profile) and *DL-Lite_{R,□}* [8,22], and \mathcal{EL} [3] with nonmonotonic negation under the EFWFS.

Definition 8 Recall that a *DL-Lite_{R,□}* knowledge base consists of a pair $(\mathcal{T}, \mathcal{A})$, where the TBox \mathcal{T} is a finite set of concept and role inclusion axioms $U_1 \sqcap \dots \sqcap U_n \sqsubseteq V$, and the ABox \mathcal{A} is a finite set of concept and role membership axioms $C(a)$ and $R(a, b)$, respectively. A *DL-Lite_{R,□,not}* knowledge base $(\mathcal{T}, \mathcal{A})$ consists of a finite set of inclusion axioms \mathcal{T} and a finite set of membership axioms \mathcal{A} , where:

- Any *DL-Lite_{R,□}* inclusion axiom is a *DL-Lite_{R,□,not}* inclusion axiom.
- If $U_1 \sqcap \dots \sqcap U_n \sqsubseteq V$ and $U'_1 \sqcap \dots \sqcap U'_m \sqsubseteq V$ with $n, m > 0$ are both either concept or role inclusion axioms in *DL-Lite_{R,□}*, and V is positive (i.e., not of the form $V = \neg V'$), then $U_1 \sqcap \dots \sqcap U_n \sqcap \text{not} U'_1 \sqcap \dots \sqcap \text{not} U'_m \sqsubseteq V$ is a *DL-Lite_{R,□,not}* concept or role inclusion axiom, respectively. Here, the U_i 's and U'_i 's contain no conjunction, and $\text{not} U'_i$ is the negation as failure (as opposed to the classical “ \neg ” in *DL-Lite*).
- For any concept A , any role R , and any individuals a, b , the expressions $A(a)$ and $R(a, b)$ are concept and role membership axioms, respectively.

A *DL-Lite_{R,□,not}* knowledge base $(\mathcal{T}, \mathcal{A})$ is a *DL-Lite_{R,not}* knowledge base iff all inclusion axioms in \mathcal{T} contain precisely one positive atom on the left-hand side.

Finally, we define $\mathcal{EL}_{\text{not}}$ as the extension of \mathcal{EL} that allows formulas of the form $\text{not} C$ for atomic concepts $C = A$ and for concepts $C = \exists R.B$ to occur in top-level conjunctions on the left-hand side of TBox-axioms. ■

The semantics of *DL-Lite_{R,not}* (resp., *DL-Lite_{R,□,not}*) is defined by translating a given *DL-Lite_{R,not}* (resp., *DL-Lite_{R,□,not}*) knowledge base KB into a normal Datalog[±] program P_{KB} and by computing the well-founded semantics of P_{KB} . The details of the translation of *DL-Lite_{R,not}* (resp., *DL-Lite_{R,□,not}*) into Datalog[±] are an extension of the translation of *DL-Lite_R* (resp., *DL-Lite_{R,□}*) given in [7]. Similarly, it is possible to translate $\mathcal{EL}_{\text{not}}$ into our formalism. Note that the sameAs(a, b) and differentFrom(a, b) constraints (specifying that the two individuals a and b are the same and different, respectively) that may be contained in a given knowledge base (over an ontology language without UNA) can be easily enforced by adding appropriate equalities $a = b$ and inequalities $\neg(a = b)$ to the guarded fixed point sentence $\text{efwfs}(P_{KB})$.

⁵ Note that although *DL-Lite_R* adopts the UNA, it actually does not require it, since making this assumption would have no impact on the semantic consequences of a *DL-Lite_R* ontology.

Example 9 (Holidays (cont’d)) Consider again Example 2. The two axioms (1) and (2) are translated into the following normal Datalog[±] rules:

$$\begin{aligned} p_{Dest}(X), p_{Swimming}(X), \neg p_{Beach}(X) &\rightarrow \exists Y. p_{BB}(X, Y); \\ p_{Dest}(X), p_{Swimming}(X), \neg p_{BB}^{\exists}(X) &\rightarrow p_{Beach}(X); \\ p_{BB}(X, Y) &\rightarrow p_{BB}^{\exists}(X), \end{aligned}$$

where $p_{BB}^{\exists}(X)$ is a new predicate for $\exists BeachBus$.

Suppose next that we are additionally given a database with holiday destinations $Dest(d_1)$, $Dest(d_2)$, and $Dest(d_3)$, which we want to be different, i.e., we assume that $\text{differentFrom}(d_1, d_2)$, $\text{differentFrom}(d_2, d_3)$, and $\text{differentFrom}(d_1, d_3)$. We want to formalize the idea that any destination where one can swim should have a beach or a bus to a location with a beach — otherwise one has to take delight in walking. This can be achieved by considering the rules (1) and (2) along with the additional rule

$$Dest \sqcap \text{not } Beach \sqcap \text{not } \exists BeachBus \sqsubseteq WalkingOnly. \quad (3)$$

Furthermore, we may assume that at any place where one is not confined to only walking, one can also swim:

$$Dest \sqcap \text{not } WalkingOnly \sqsubseteq Swimming. \quad (4)$$

Consider the following further facts: $WalkingOnly(d_1)$ and $BeachBus(d_2, d_3)$. Clearly, $WalkingOnly(d_1)$ implies that $Swimming(d_1)$ cannot be derived — because rule (4), the only rule that can derive $Swimming(d_1)$, requires the negation of $WalkingOnly(d_1)$ to be true. Thus, the WFS of the knowledge base includes $\text{not } Swimming(d_1)$. This is in contrast to what would happen if we interpreted not as “classical” negation: in the latter case, we could not derive anything from $WalkingOnly(d_1)$, as axiom (4) would trivially be satisfied for d_1 . Other facts that are derived for d_1 are $\text{not } Beach(d_1)$ and $\text{not } \exists BeachBus(d_1)$ ($= \text{not } R(x, y)$ for any y), because the fact $\text{not } Swimming(d_1)$ implies that rules (1) and (2) cannot be used to derive $Beach(d_1)$ or $\exists BeachBus(d_1)$ (note that we use the fact that d_1 has to be different from d_2 and d_3 , because of our ABox assumptions). Concerning the destination d_2 , the WFS contains the following atoms: $\text{not } Beach(d_2)$, $\text{not } WalkingOnly(d_2)$, and $Swimming(d_2)$. ■

The complexity of answering covered NBCQs for any of our $DL\text{-Lite}_{\text{not}}$ logics and for $\mathcal{EL}_{\text{not}}$ can now be determined using Theorem 7. Note that Theorem 10 also yields immediate bounds for the complexity of standard DL problems such as instance and satisfiability checking.

Theorem 10 *Let \mathcal{L} be any of $DL\text{-Lite}_{\mathcal{R}, \sqcap, \text{not}}$, $DL\text{-Lite}_{\mathcal{R}, \text{not}}$ or $\mathcal{EL}_{\text{not}}$. Then, given a knowledge base $KB = (\mathcal{T}, \mathcal{A})$ and an acyclic (resp., general) covered NBCQ Q , deciding whether Q is true under the EFWS of KB is in EXPTIME (resp., 2-EXPTIME) for combined complexity.*

7 Related Work

There is already a substantial amount of work on combining rules and ontologies. The main direction of research so far has been to combine rules and ontologies into dl-programs consisting of a knowledge base together with a set of rules. This combination can be carried out in a loose or tight fashion. Representatives of the former are in particular the dl-programs in [10]. The hybrid MKNF knowledge bases in [21] are also close in spirit. Some representatives of tight integrations of rules and ontologies include the works by Rosati [23] and Lukasiewicz [18]. SWRL [15] and WRL [2] also belong to this category. For several of the above combinations of rules and ontologies, a well-founded semantics has been defined: [11], [19], [17], and [9] define a well-founded semantics for the loosely integrated dl-programs in [10], for the tightly integrated dl-programs in [18], for the hybrid MKNF knowledge bases in [21], and for an integration of rules and ontologies that is close in spirit to Rosati’s approach [23], respectively.

We achieve the combination of rules and ontologies by a reduction from description logics to logic programming formalisms. Obviously our work is based on the earlier work on Datalog[±]. Based on the same idea of translating ontologies into logic programming rules and hence closely related to our work are [1], [24], [14], and [16]. Probably the closest relationship to our work has the paper on FDNC-rules by [12] where the stable semantics is used in order to obtain a rule-based formalism with negation-as-failure that allows for the formulation of ontological knowledge.

8 Summary and Outlook

We have defined the equality-friendly WFS for Datalog with existentially quantified variables in rule heads and negations in rule bodies. Via a translation of its guarded fragment to *guarded fixed point logic*, we have then proved the decidability in the guarded case, and obtained complexity results for this case. These are important contributions in their own right. In addition, since the approach does not make the UNA, it can be readily used to extend DLs by nonmonotonic negation under the WFS, as illustrated along several DLs, including *DL-Lite_R*, underlying the important OWL 2 QL profile.

Interesting topics for future research include to investigate the data complexity of guarded normal Datalog[±] and to explore how the approach can be extended by keys and to other ontology languages, including OWL 2 EL and OWL 2 RL.

Acknowledgments. This work was supported by the EPSRC grant EP/H051511/1 “Ex-ODA: Integrating Description Logics and Database Technologies for Expressive Ontology-Based Data Access”, by the European Research Council under the EU’s 7th Framework Programme (FP7/2007-2013)/ERC grant 246858 (DIADEM), by the European Commission under the EU’s FP7 grant 238975 (SEALS), and by a Yahoo! Research Fellowship. Georg Gottlob is a James Martin Senior Fellow, and also gratefully acknowledges a Royal Society Wolfson Research Merit Award. The work was carried out in the context of the James Martin Institute for the Future of Computing.

References

1. Alsaç, G., Baral, C.: Reasoning in description logics using declarative logic programming. Technical report, Dep. of Computer Science and Engineering, Arizona State Univ. (2001)

2. Angele, J., Boley, H., de Bruijn, J., Fensel, D., Hitzler, P., Kifer, M., Krummenacher, R., Lausen, H., Polleres, A., Studer, R.: Web Rule Language (WRL) (Sep 2005), W3C Member Submission. Available at <http://www.w3.org/Submission/WRL/>.
3. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Proc. IJCAI-2005.
4. Baral, C., Subrahmanian, V.S.: Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *J. Autom. Reasoning* 10(3), 399–420 (1993)
5. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. ICALP-1981.
6. Cali, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. KR-2008. Revised version: <http://www.dbai.tuwien.ac.at/staff/gottlob/CGK.pdf>.
7. Cali, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: *J. Web Sem.* (2012)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning* 39(3), 385–429 (2007)
9. Drabent, W., Małuszyński, J.: Well-founded semantics for hybrid rules. In: Proc. RR-2007.
10. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. *Artif. Intell.* 172(12/13), 1495–1539 (2008)
11. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Well-founded semantics for description logic programs in the Semantic Web. In: Proc. RuleML-2004.
12. Eiter, T., Simkus, M.: FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Trans. Comput. Log.* 11(2) (2010)
13. Grädel, E., Walukiewicz, I.: Guarded fixed point logic. In: Proc. LICS-1999.
14. Heymans, S., Vermeir, D.: Integrating Semantic Web reasoning and answer set programming. In: Proc. ASP-2003.
15. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Groszof, B., Dean, M.: SWRL: A Semantic Web rule language combining OWL and RuleML (May 2004), W3C Member Submission. Available at <http://www.w3.org/Submission/SWRL/>.
16. Hustadt, U., Motik, B., Sattler, U.: Reducing SHIQ-description logic to disjunctive Datalog programs. In: Proc. KR-2004.
17. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. *Artif. Intell.* 175(9/10), 1528–1554 (2011)
18. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the Semantic Web. In: Proc. ESWC-2007.
19. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the Semantic Web. *IEEE Trans. Knowl. Data Eng.* 22(11), 1577–1592 (2010)
20. Motik, B., Horrocks, I., Rosati, R., Sattler, U.: Can OWL and logic programming live together happily ever after? In: Proc. ISWC-2006.
21. Motik, B., Rosati, R.: A faithful integration of description logics with logic programming. In: Proc. IJCAI-2007.
22. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. *J. Data Semantics* 10, 133–173 (2008)
23. Rosati, R.: $\mathcal{DL}+log$: Tight integration of description logics and disjunctive Datalog. In: Proc. KR-2006.
24. Swift, T.: Deduction in ontologies via ASP. In: Proc. LPNMR-2004.
25. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. *J. ACM* 38(3), 620–650 (1991)

Finite Model Reasoning in *DL-Lite* with Cardinality Constraints^{*}

(Preliminary Results)

Yazmín Ibáñez-García

KRDB Research Centre
Free University of Bozen-Bolzano, Italy
ibanezgarcia@inf.unibz.it

1 Introduction

The relationship of description logics (DLs) and conceptual modelling has been extensively studied in the literature [5, 4, 1]. One of the advantages of using description logics as modelling languages is that along with their capability of representing knowledge they provide also reasoning services. More precisely, a conceptual model can be represented by a DL ontology (TBox), and standard reasoning services (e.g., satisfiability and subsumption) allow to verify some properties of the conceptual model (e.g., consistency) and infer relations between concepts (IS-A relationships between classes or entities) that are not explicitly expressed. In order to use DLs effectively for conceptual modelling we need to ensure (1) that the chosen DL language is expressive enough to capture faithfully the intended semantics of traditional modelling languages (e.g., UML class diagrams, ER schema), and (2) that the complexity of reasoning in the chosen DL is acceptable (e.g., tractable). Regarding (1), it is worth noticing that the domain of interest in most applications is finite, therefore, reasoning on conceptual models should be understood as *reasoning w.r.t. finite models*. The latter is not the usual assumption in DLs mainly because traditional description logics enjoy the finite model property (FMP), and hence there is no need to distinguish between reasoning w.r.t. arbitrary models, and w.r.t. finite ones. Notably, *ALC* (one of the traditional DLs) is not expressive enough for capturing *cardinality constraints*. In DLs cardinality constraints are expressed by (qualified) number restrictions. *ALCQI* –which extends *ALC* with qualified number restrictions and inverse roles– captures the semantics of UML class diagrams [4]. However, this extension of *ALC* does not enjoy the FMP any more. One drawback for the use of *ALCQI* is the complexity of reasoning: finite satisfiability of *ALCQI* knowledge bases is EXPTIME-complete [11]. The high complexity of reasoning makes *ALCQI* not very attractive for the conceptual modelling task; specially because no *optimized algorithms* for finite model reasoning exist. As an alternative, members of the *DL-Lite*-family of description logics including unqualified

^{*} We would like to thank the anonymous reviewers, as well as Alessandro Artale, André Hernich, and Víctor Gutiérrez-Basulto for valuable remarks to improve the final version of this paper.

number restrictions [2] capture relevant modelling features [1]. However, little has been done in the study of the complexity of reasoning w.r.t. finite models in *DL-Lite* [13]. A consideration around the use of number restrictions (qualified or unqualified) regards their semantics: on the DL side, number restrictions, intended for possible infinite model semantics, constraint the numbers of objects that are related to a certain object; while cardinality constraints in conceptual modelling, intended for finite model semantics, establish relationships among the cardinality of classes/entities.

The purpose of this paper is to bring attention to finite model reasoning in description logics from a model theoretical view point. We adapt existing techniques [6] and show that the complexity of finite model reasoning in the Horn fragment of *DL-Lite* is tractable when only global functionality constraints are considered. While this result seems to be an almost straightforward consequence of existing results [13]; the approach taken in this paper leads to a deeper understanding of the structural properties of finite models for *DL-Lite* knowledge bases. We also observe that when allowing the use of arbitrary cardinality constraints, finite satisfiability becomes harder than arbitrary reasoning in $DL-Lite_{horn}^N$. In Section 5 we provide an intuition for an upper bound on the complexity of finite model reasoning in $DL-Lite_{horn}^N$. The results and observations presented in this paper shall serve as the foundation for future work on the finite model theory in light weight description logics [2, 3].

2 Preliminaries

DL-Lite syntax and semantics The language of $DL-Lite_{horn}^N$ contains *individual names* a_0, a_1, \dots , *concept names* A_0, A_1, \dots , *role names* P_0, P_1, \dots . *Complex roles* R , and *concepts* B are built according to the following syntax rule:

$$R ::= P_i \mid P_i^-, \quad B ::= \perp \mid A_i \mid \geq n R,$$

where n in *number restrictions* ($\geq n R$) is a positive integer. We call *existentials* those number restrictions with $n = 1$, denoted also by $\exists R$. A $DL-Lite_{horn}^N$ -TBox \mathcal{T} is a finite set of axioms of the form $B_1 \sqcap \dots \sqcap B_k \sqsubseteq B$, $k \geq 0$, where by definition the empty conjunction is \top . We also consider the sublogic $DL-Lite_{horn}^F$, which of all number restrictions only allows for existentials, and those with $n = 2$ occurring only in concept inclusions of the form $\geq 2 R \sqsubseteq \perp$, which are called *global functionality constraints*, and are denoted by $(\text{funct } R)$. An ABox \mathcal{A} is a finite set of assertions of the form: $A(a_i)$ or $P(a_i, a_j)$. Together, a TBox \mathcal{T} and an ABox \mathcal{A} constitute a $DL-Lite_{horn}^F$ *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. We use $\text{ind}(\mathcal{A})$ to denote the set of individual names occurring in \mathcal{A} ; $\text{role}(\mathcal{K})$ the set of role names in \mathcal{K} , and $\text{role}^\pm(\mathcal{K})$ the set of roles $\{P_k, P_k^- \mid P_k \in \text{role}(\mathcal{K})\}$. For a role $R \in \text{role}^\pm(\mathcal{K})$, $R^- = P_k$ if $R = P_k^-$, and $R^- = P_k^-$ if $R = P_k$. Finally, $\text{concepts}(\mathcal{K})$ denotes the set of basic concepts occurring in \mathcal{K} , and $\text{concepts}(\mathcal{T})$, for those occurring in \mathcal{T} .

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty *domain* $\Delta^{\mathcal{I}}$, and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each individual name a_i an element

$a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$; to each concept name A_j a subset $A_j^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and to each role name P_k , a binary relation $P_k^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concepts and roles as follows:

$$\begin{aligned}
(P_k^-)^{\mathcal{I}} &= \{(e, d) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d, e) \in P_k^{\mathcal{I}}\}; & (\text{inverse role}) \\
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}}; & (\text{Top}) \\
\perp^{\mathcal{I}} &= \emptyset; & (\text{Bottom}) \\
(\geq n R)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \#\{e \in \Delta^{\mathcal{I}} \mid (d, e) \in R^{\mathcal{I}}\} \geq n\}; & (\text{number rest.}) \\
(B_i \sqcap B_j)^{\mathcal{I}} &= B_i^{\mathcal{I}} \cap B_j^{\mathcal{I}}; & (\text{conjunction})
\end{aligned}$$

where $\#$ denotes the cardinality of a set. An interpretation \mathcal{I} *satisfies* a TBox axiom $\prod_k B_k \sqsubseteq B$ iff $(\prod_k B_k)^{\mathcal{I}} \subseteq B^{\mathcal{I}}$, in that case we write $\mathcal{I} \models \prod_k B_k \sqsubseteq B$; similarly, $\mathcal{I} \models (\text{funct } R)$ iff whenever both $(d, e) \in R^{\mathcal{I}}$ and $(d, e') \in R^{\mathcal{I}}$, then $e = e'$. For ABox assertions we have that $\mathcal{I} \models A(a_i)$ iff $a_i^{\mathcal{I}} \in A^{\mathcal{I}}$; and $\mathcal{I} \models P_k(a_i, a_j)$ iff $(a_i^{\mathcal{I}}, a_j^{\mathcal{I}}) \in P_k^{\mathcal{I}}$. A knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is *satisfiable* (or consistent) if there is an interpretation \mathcal{I} , satisfying every axiom in \mathcal{T} and every assertion in \mathcal{A} . In this case we write $\mathcal{I} \models \mathcal{K}$ (as well as $\mathcal{I} \models \mathcal{T}$, and $\mathcal{I} \models \mathcal{A}$), and we say that \mathcal{I} is a *model* of \mathcal{K} (and of \mathcal{T} and \mathcal{A}). If \mathcal{I} is finite (i.e., its domain is finite) we say that a \mathcal{K} (as well as \mathcal{T} and \mathcal{A}) is *finitely satisfiable*. The *type* of d in \mathcal{I} is the set $t_{\mathcal{I}}(d) = \{B \mid d \in B^{\mathcal{I}}\}$, where B is a $DL\text{-Lite}_{horn}^{\mathcal{N}}$ -concept. The set of all types of \mathcal{I} , is $types(\mathcal{I}) = \{t_{\mathcal{I}}(d) \mid d \in \Delta^{\mathcal{I}}\}$. We consider standard reasoning tasks. Specifically, *satisfiability* and *subsumption*. Let $\mathcal{L} \in \{DL\text{-Lite}_{horn}^{\mathcal{F}}, DL\text{-Lite}_{horn}^{\mathcal{N}}\}$. The *satisfiability problem* consists on deciding, given an \mathcal{L} -KB \mathcal{K} , whether \mathcal{K} is satisfiable; while the *subsumption problem* amounts to decide, given an \mathcal{L} -TBox \mathcal{T} and \mathcal{L} -concepts C_1 and C_2 , whether $\mathcal{T} \models C_1 \sqsubseteq C_2$, i.e., whether $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ in every model \mathcal{I} of \mathcal{T} .

3 Model Theoretical Characterizations

Lutz et. al., [10] provide a model theoretical characterization of $DL\text{-Lite}_{horn}$ (without number restrictions) based on (*equi*)*simulation*, a weaker notion of the classical (bi)simulation [7]. In order to capture the *counting* capability of $DL\text{-Lite}_{horn}^{\mathcal{N}}$ we extend this notion similarly to the graded-bisimulation in [12].

For a $DL\text{-Lite}_{horn}^{\mathcal{N}}$ interpretation \mathcal{I} , an object $d \in \Delta^{\mathcal{I}}$, and a role R ,

$$R\text{-succ}^{\mathcal{I}}(d) = \{e \in \Delta^{\mathcal{I}} \mid (d, e) \in R^{\mathcal{I}}\}$$

is the set of *R-successors* of d in \mathcal{I} .

Let $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \mathcal{I}_1)$ and $\mathcal{I}_2 = (\Delta^{\mathcal{I}_2}, \mathcal{I}_2)$ be two $DL\text{-Lite}_{horn}^{\mathcal{N}}$ interpretations. A *graded equisimulation* (or *g-equisimulation*) between \mathcal{I}_1 and \mathcal{I}_2 is a relation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ that satisfies the following properties:

- (atom)** for every concept name A , if $(d, e) \in \rho$ then $d \in A^{\mathcal{I}_1}$ iff $e \in A^{\mathcal{I}_2}$;
- (role)** for every role R , if $(d, e) \in \rho$, then the following hold:
 - (i) for every finite set $S \subseteq R\text{-succ}^{\mathcal{I}_1}(d)$, there exists a finite set $S' \subseteq R\text{-succ}^{\mathcal{I}_2}(e)$, such that $\#S = \#S'$; and

- (ii) for every finite set $S \subseteq R\text{-succ}^{\mathcal{I}_2}(e)$, there exists a finite set $S' \subseteq R\text{-succ}^{\mathcal{I}_1}(d)$, such that $\#S = \#S'$.¹

ρ is called *global* if and only if (i) for every $d \in \Delta^{\mathcal{I}_1}$ there is some $e \in \Delta^{\mathcal{I}_2}$ with $(d, e) \in \rho$, (ii) for every $e \in \Delta^{\mathcal{I}_2}$ there is some $d \in \Delta^{\mathcal{I}_1}$ with $(d, e) \in \rho$.

We write $(\mathcal{I}_1, d) \approx (\mathcal{I}_2, e)$ if there exists a g -equisimulation ρ between \mathcal{I}_1 and \mathcal{I}_2 such that $(d, e) \in \rho$. Finally, we say that \mathcal{I}_1 is *g -equisimilar* to \mathcal{I}_2 , denoted as $\mathcal{I}_1 \approx \mathcal{I}_2$, if there is a *global g -equisimulation* ρ between \mathcal{I}_1 and \mathcal{I}_2 .

Lemma 1. *Let \mathcal{T} be a $DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ TBox, C a $DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ concept; and \mathcal{I}_1 and \mathcal{I}_2 be two $DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ interpretations over the signature of \mathcal{T} and C . The following statements hold:*

- (a) *$DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ concepts are invariant under g -equisimulations: $(\mathcal{I}_1, d) \approx (\mathcal{I}_2, e)$ implies $d \in C^{\mathcal{I}_1}$ iff $e \in C^{\mathcal{I}_2}$.*
- (b) *$DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ TBoxes are invariant under global g -equisimulations: if $\mathcal{I}_1 \approx \mathcal{I}_2$ then $\mathcal{I}_1 \models \mathcal{T}$ iff $\mathcal{I}_2 \models \mathcal{T}$.*
- (c) *Every model of a $DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ TBox is g -equisimilar to a tree-shaped model.*

Canonical Models We use a standard characterization of unrestricted entailment in terms of canonical models [8]. A canonical interpretation for a $DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is constructed by (i) expanding the set of individual names in \mathcal{A} with an additional set of individuals $\{d_R \mid R \in \text{role}^{\pm}(\mathcal{T})\}$ that serve as witness of existentials, and (ii) expanding the extensions of concept and role names as required by \mathcal{T} . A role R is called *generating* in \mathcal{K} if there exist $a \in \text{ind}(\mathcal{A})$ and $R_0, \dots, R_n = R$ such that the following conditions hold:

- (agen) $\mathcal{K} \models \exists R_0(a)$ but $R_0(a, b) \notin \mathcal{A}$ for all $b \in \text{ind}(\mathcal{A})$ (written $a \rightsquigarrow d_{R_0^-}$).
- (rgen) For $i < n$, $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_i^- \neq R_{i+1}$ (written $d_{R_i^-} \rightsquigarrow d_{R_{i+1}^-}$).

Definition 1. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ KB. The canonical interpretation $\mathcal{I}_{\mathcal{K}} = (\Delta^{\mathcal{I}_{\mathcal{K}}}, \cdot^{\mathcal{I}_{\mathcal{K}}})$ of \mathcal{K} is defined as follows:*

$$\begin{aligned} \Delta^{\mathcal{I}_{\mathcal{K}}} &= \text{ind}(\mathcal{A}) \cup \{d_R \mid R^- \text{ is generating in } \mathcal{K}\}; \\ a^{\mathcal{I}_{\mathcal{K}}} &= a \text{ for every } a \in \text{ind}(\mathcal{A}); \\ A^{\mathcal{I}_{\mathcal{K}}} &= \{a \in \text{ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup \{d_R \in \Delta^{\mathcal{I}_{\mathcal{K}}} \mid \mathcal{T} \models \exists R \sqsubseteq A\}; \\ P^{\mathcal{I}_{\mathcal{K}}} &= \{(a_i, a_j) \in \text{ind}(\mathcal{A}) \times \text{ind}(\mathcal{A}) \mid P(a_i, a_j) \in \mathcal{A}\} \cup \\ &\quad \{(a, d_{P^-}) \mid a \rightsquigarrow d_{P^-}\} \cup \{(d_P, a) \mid a \rightsquigarrow d_P\} \cup \\ &\quad \{(d_S, d_{P^-}) \mid d_S \rightsquigarrow d_{P^-}\} \cup \{(d_P, d_S) \mid d_S \rightsquigarrow d_P\}. \end{aligned}$$

The canonical interpretation $\mathcal{I}_{\mathcal{K}}$ of a given KB \mathcal{K} can be computed in polynomial time on the size of \mathcal{K} [8], and serves as a finite compact representation of every model of \mathcal{K} . However, $\mathcal{I}_{\mathcal{K}}$ is *not* itself in general a model of \mathcal{K} , as the following example shows:

¹ Clearly, if both $R\text{-succ}^{\mathcal{I}_1}(d)$ and $R\text{-succ}^{\mathcal{I}_2}(e)$ are finite, these conditions are equivalent to $\#R\text{-succ}^{\mathcal{I}_1}(d) = \#R\text{-succ}^{\mathcal{I}_2}(e)$.

Example 1. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T} = \{\exists S \sqsubseteq \exists P_1, \exists P_1^- \sqsubseteq \exists P_2, \exists P_2^- \sqsubseteq \exists P_1, \exists P_2^- \sqsubseteq \exists P_3^-, \exists P_3 \sqsubseteq \exists S^-(\text{func} P_1^-), (\text{func} P_2^-), (\text{func} S), B \sqsubseteq \exists P_1\}$, and $\mathcal{A} = \{B(a)\}$.

The canonical interpretation $\mathcal{I}_{\mathcal{K}}$, depicted in Figure 1, clearly violates the functionality of P_1^- and hence is not a model of \mathcal{K} .

In general, $\mathcal{I}_{\mathcal{K}}$ cannot be a model of \mathcal{K} since it is finite, and $DL\text{-Lite}_{horn}^{\mathcal{N}}$ does not enjoy the finite model property (FMP). A standard way to construct a (canonical) model from $\mathcal{I}_{\mathcal{K}}$ is to *unravel* it into a forest-shaped interpretation $\mathcal{U}_{\mathcal{K}}$ [2, 8]. We omit the definition of $\mathcal{U}_{\mathcal{K}}$ here and focus only in its properties:

Lemma 2. *Let \mathcal{K} be a $DL\text{-Lite}_{horn}^{\mathcal{N}}$ knowledge base, and $\mathcal{U}_{\mathcal{K}}$ the unravelling of the canonical interpretation $\mathcal{I}_{\mathcal{K}}$, then the following hold:*

- (p1) \mathcal{K} is satisfiable iff $\mathcal{U}_{\mathcal{K}} \models \mathcal{K}$.
- (p2) For every $DL\text{-Lite}_{horn}^{\mathcal{N}}$ TBox axiom φ , $\mathcal{K} \models \varphi$ iff $\mathcal{U}_{\mathcal{K}} \models \varphi$.

4 Finite Model Reasoning in $DL\text{-Lite}_{horn}^{\mathcal{F}}$

In this section, we study finite model reasoning in $DL\text{-Lite}_{horn}^{\mathcal{F}}$. Notably, the FMP it is already lost when considering only functionality constraints. Let us take the following $DL\text{-Lite}_{horn}^{\mathcal{F}}$ KB to illustrate this:

$$\mathcal{K}' = (\mathcal{T} \cup \{B \sqcap \exists P_2^- \sqsubseteq \perp\}, \mathcal{A}) \quad (1)$$

with \mathcal{T} and \mathcal{A} from Example 1. It is not hard to see that \mathcal{K}' is satisfiable only by infinite models. Intuitively, in every model \mathcal{I} of \mathcal{K}' , there is an infinite sequence of objects connected by P_1 and P_2 starting from $a^{\mathcal{I}}$: since a is an instance of B , $a^{\mathcal{I}}$ has a P_1 -successor, d_1 , and from $\exists P_1^- \sqsubseteq \exists P_2$, d_1 has a P_2 successor different from $a^{\mathcal{I}}$ (from $B \sqcap \exists P_2^- \sqsubseteq \perp$), say d_2 , from $\exists P_2^- \sqsubseteq \exists P_1$, d_2 has a P_1 -successor, d_3 , different from d_1 , (since P_1^- is functional), and d_3 has a P_2 -successor, d_4 , different from d_2 (since P_2^- is functional). These arguments can be used repeatedly to see that indeed an infinite number of objects are needed to satisfy the constraints in \mathcal{K}' .

In order to provide a method for reasoning in $DL\text{-Lite}_{horn}^{\mathcal{F}}$ w.r.t. finite models, we follow the approach taken by Cosmadakis et. al., [6] for characterizing finite implication of unary inclusion dependencies (UINDS) and functionality dependencies in databases. Given a $DL\text{-Lite}_{horn}^{\mathcal{F}}$ -KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we show that it is possible to ‘enrich’ \mathcal{T} in such a way that it explicitly contains concept inclusions and functionality constraints that hold in every finite model of \mathcal{T} . We adapt the idea behind the axiomatization presented by Cosmadakis et. al., [6], and define a closure of a given TBox \mathcal{T} in terms of arbitrary reasoning. Differently from what is done by Rosati [13], we do not exclude disjointness axioms of the form $B_1 \sqcap \dots \sqcap B_k \sqsubseteq \perp$ from \mathcal{T} for defining such a closure.

To simplify the presentation we consider an extension of $DL\text{-Lite}_{horn}^{\mathcal{F}}$ with axioms of the form $B_i \geq B_j$, with the following intended semantics for finite models: a finite interpretation \mathcal{I} satisfies $B_i \geq B_j$ if and only if $\#(B_i)^{\mathcal{I}} \geq \#(B_j)^{\mathcal{I}}$.

Definition 2. For a given $DL\text{-Lite}_{\text{horn}}^{\mathcal{F}}$ TBox \mathcal{T} , $\text{finClosure}(\mathcal{T})$ denotes the minimal set of axioms satisfying the following conditions:

1. $\mathcal{T} \subseteq \text{finClosure}(\mathcal{T})$;
2. For every pair of basic concepts B_1, B_2 occurring in \mathcal{T} . If $\text{finClosure}(\mathcal{T}) \models B_1 \sqsubseteq B_2$, then $B_2 \geq B_1 \in \text{finClosure}(\mathcal{T})$;
3. if $(\text{funct } R) \in \text{finClosure}(\mathcal{T})$ then $\exists R \geq \exists R^- \in \text{finClosure}(\mathcal{T})$;
4. if $\{B_1 \geq B_2, B_2 \geq B_3\} \subseteq \text{finClosure}(\mathcal{T})$ then $B_1 \geq B_3 \in \text{finClosure}(\mathcal{T})$;
5. if $\{(\text{funct } R), \exists R^- \geq \exists R\} \subseteq \text{finClosure}(\mathcal{T})$ then $(\text{funct } R^-) \in \text{finClosure}(\mathcal{T})$;
6. if $\text{finClosure}(\mathcal{T}) \models B_1 \sqsubseteq B_2$, and $B_1 \geq B_2 \in \text{finClosure}(\mathcal{T})$ then $B_2 \sqsubseteq B_1 \in \text{finClosure}(\mathcal{T})$.

From 1, it follows that every model of $\text{finClosure}(\mathcal{T})$ is also a model of \mathcal{T} . Since TBox reasoning in $DL\text{-Lite}_{\text{horn}}^{\mathcal{F}}$ is PTIME-complete [2], the following holds:

Proposition 1. $\text{finClosure}(\mathcal{T})$ can be computed in polynomial time on the size of \mathcal{T} .

(1)-(4) in Definition 2 are based in logical consequences and are therefore sound w.r.t. arbitrary models. (5) and (6), on the other hand, are not sound w.r.t. infinite models, but a simple counting argument shows that they are sound w.r.t. finite models. Hence, we have the following result:

Lemma 3. Let \mathcal{T} be a $DL\text{-Lite}_{\text{horn}}^{\mathcal{F}}$ -TBox. Then, the following hold:

- (a) if $\text{finClosure}(\mathcal{T}) \models \prod_k B_k \sqsubseteq B$ then $\mathcal{T} \models_{\text{fin}} \prod_k B_k \sqsubseteq B$;
- (b) if $(\text{funct } R) \in \text{finClosure}(\mathcal{T})$ then $\mathcal{T} \models_{\text{fin}} (\text{funct } R)$.

Moreover, the introduction of axioms of the form $B_i \geq B_j$, induces a directed graph $(\mathcal{V}, \mathcal{E})$, with \mathcal{V} the set of concepts occurring in \mathcal{T} and $(B_i, B_j) \in \mathcal{E}$ iff $B_i \geq B_j \in \text{finClosure}(\mathcal{T})$. The implications w.r.t. finite models can be better understood by observing the structure of $(\mathcal{V}, \mathcal{E})$.

Example 2 (finClosure). Consider the TBox \mathcal{T} from Example 1. $\text{finClosure}(\mathcal{T})$ contains (among others) the axioms $\mathcal{T}_1 = \{\exists P_2 \sqsubseteq \exists P_1^-, \exists P_1 \sqsubseteq \exists P_2^-, (\text{funct } P_1), (\text{funct } P_2)\}$. Figure 2 shows a portion of the graph induced by $\text{finClosure}(\mathcal{T})$. The dashed lines represent ‘ \geq ’ inferred by concept inclusions, and the solid lines are ‘ \geq ’ introduced by functionality assertions (rules 2–4). From a solid (dashed) edge (B_i, B_j) belonging to a cycle, it is inferred a solid (dashed) edge (B_j, B_i) (rules 5–6). In the example, from the edge $(d_{P_1}, d_{P_2^-})$, corresponding to the axiom $\exists P_2^- \sqsubseteq \exists P_1$, it is inferred that $\exists P_1 \sqsubseteq \exists P_2^- \in \text{finClosure}(\mathcal{T})$. Analogously, from the solid line labelled with P_1^- , corresponding to $(\text{funct } P_1^-)$ it is inferred that $(\text{funct } P_1)$.

If there is an unsatisfiable concept B_i , this is reflected by an axiom of the form $\perp \geq B_i$. Let us consider \mathcal{K}' from (1). We have that from the ‘cycle rules’, $\exists P_1 \sqsubseteq \exists P_2^- \in \text{finClosure}(\mathcal{T}')$. Hence, $\text{finClosure}(\mathcal{T}') \models \{\exists P_1 \sqsubseteq \exists P_2^-, B \sqsubseteq \exists P_1, B \sqcap \exists P_2^- \sqsubseteq \perp\}$, which implies that $\text{finClosure}(\mathcal{T}') \models B \sqsubseteq \perp$, and then, by rule 1, $\perp \geq B \in \text{finClosure}(\mathcal{T}')$ (see Figure 3). This means that in every finite model \mathcal{I} of \mathcal{T}' , $\sharp B^{\mathcal{I}} = 0$. An inconsistency w.r.t. finite models is then derived from the ABox assertion $B(a)$.

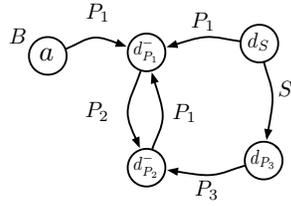


Fig. 1: $\mathcal{I}_{\mathcal{K}}$ with $\mathcal{K} = (\mathcal{T}, \mathcal{A})$

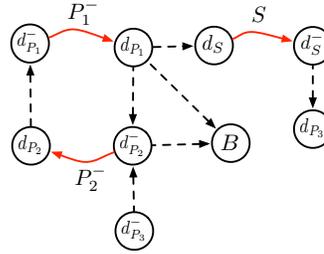


Fig. 2: $\text{finClosure}(\mathcal{T})$

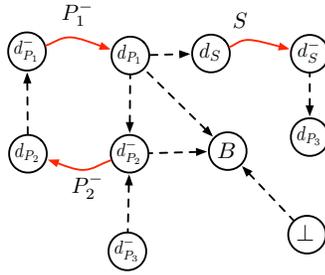


Fig. 3: $\text{finClosure}(\mathcal{T}')$

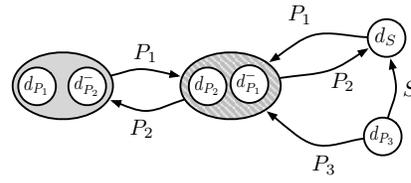


Fig. 4: $\mathcal{I}_{\hat{\tau}}$

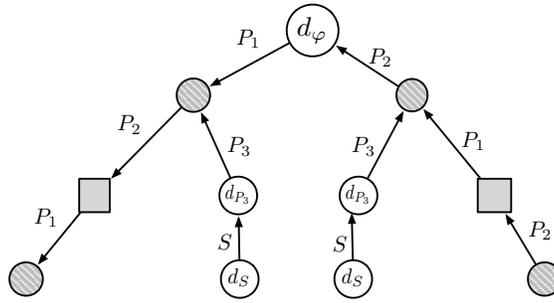


Fig. 5: $\mathcal{U}_{\hat{\tau}}$

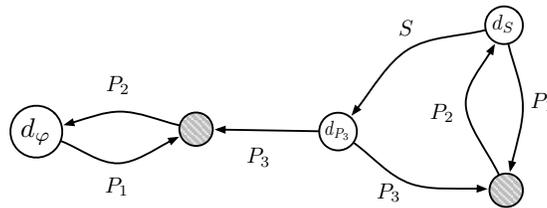


Fig. 6: $\mathcal{I}_{\hat{\tau}}^f$

We proceed to prove that finClosure is complete. More precisely, we show the following:

Lemma 4. *Let \mathcal{T} be a $DL\text{-Lite}_{horn}^{\mathcal{F}}$ -TBox, and φ a $DL\text{-Lite}_{horn}^{\mathcal{F}}$ axiom on the signature of \mathcal{T} , i.e., φ is either a concept inclusion or a functionality constraint. If $\mathcal{T} \models_{\text{fin}} \varphi$ then $\text{finClosure}(\mathcal{T}) \models \varphi$.*

Proof. We shall show that $\text{finClosure}(\mathcal{T}) \not\models \varphi$ implies $\mathcal{T} \not\models_{\text{fin}} \varphi$. In what follows, we fix a $DL\text{-Lite}_{horn}^{\mathcal{F}}$ -TBox \mathcal{T} , and set $\widehat{\mathcal{T}} = \text{finClosure}(\mathcal{T})$. Then, by Lemma 2-(p2), it suffices to show that $\mathcal{U}_{\widehat{\mathcal{T}}} \not\models \varphi$ implies $\mathcal{T} \not\models_{\text{fin}} \varphi$, where $\mathcal{U}_{\widehat{\mathcal{T}}}$ is the unravelling of a canonical interpretation $\mathcal{I}_{\widehat{\mathcal{T}}}$ that depends on φ . Specifically, if $\varphi = C \sqsubseteq B$, then the ‘root’ of $\mathcal{U}_{\widehat{\mathcal{T}}}$ is an object $d \in C^{\mathcal{U}_{\widehat{\mathcal{T}}}} \setminus B^{\mathcal{U}_{\widehat{\mathcal{T}}}}$. On the other hand, if $\varphi = (\text{funct } R)$, then $\mathcal{U}_{\widehat{\mathcal{T}}}$ is rooted at an object d with two different R -successors e and e' .

Let us consider the case $\varphi = C \sqsubseteq B$. We construct a finite model $\mathcal{I}_{\mathcal{T}}^f$ of \mathcal{T} such that $\mathcal{U}_{\widehat{\mathcal{T}}} \not\models C \sqsubseteq B$ implies $\mathcal{I}_{\mathcal{T}}^f \not\models C \sqsubseteq B$. But first, we introduce some useful notation. For any two concepts B_1, B_2 , we write $B_1 \sqsubseteq_{\widehat{\mathcal{T}}} B_2$ whenever $\widehat{\mathcal{T}} \models B_1 \sqsubseteq B_2$; and $B_1 \equiv_{\widehat{\mathcal{T}}} B_2$, if additionally $B_2 \sqsubseteq_{\widehat{\mathcal{T}}} B_1$. Since $\equiv_{\widehat{\mathcal{T}}}$ is an equivalence relation, the set of concepts $\mathfrak{C} = \{\exists R \mid R \in \text{role}^{\pm}(\widehat{\mathcal{T}})\}$ can be partitioned into equivalence classes w.r.t. $\equiv_{\widehat{\mathcal{T}}}$. Then, $[\exists R] \in \mathfrak{C} / \equiv_{\widehat{\mathcal{T}}}$ denotes the following equivalence class of concepts: $[\exists R] = \{B_i \in \mathfrak{C} \mid B_i \equiv_{\widehat{\mathcal{T}}} \exists R\}$. Before moving forward with the definition of $\mathcal{I}_{\mathcal{T}}^f$, we observe that the canonical interpretation $\mathcal{I}_{\widehat{\mathcal{T}}}$ constructed as in Definition 1 may introduce multiple witnesses for a given existential. We set $d_s = d'_s$ whenever $\exists S \equiv_{\widehat{\mathcal{T}}} \exists S'$. Therefore, domain of the canonical interpretation $\mathcal{I}_{\widehat{\mathcal{T}}}$ contains exactly one element d_S for each class $[\exists S] \in \mathfrak{C} / \equiv_{\widehat{\mathcal{T}}}$ (e.g., as in Figure 4).

We write $[\exists S_i] \xrightarrow{R} [\exists S_j]$ iff $\exists S_i \sqsubseteq_{\widehat{\mathcal{T}}} \exists R$ and $\exists R^- \in [\exists S_j]$. Analogously, $\exists S_i \geq_{\widehat{\mathcal{T}}} \exists S_j$ denotes that $\exists S_i \geq \exists S_j \in \widehat{\mathcal{T}}$.

We observe that $\geq_{\widehat{\mathcal{T}}}$ induces a coarser partition on \mathfrak{C} . For a concept $\exists S_i \in \mathfrak{C}$, the *cluster* of $\exists S_i$ is the set $\mathfrak{C}(\exists S_i) = \{\exists S_j \in \mathfrak{C} \mid \exists S_i \geq_{\widehat{\mathcal{T}}} \exists S_j \text{ and } \exists S_j \geq_{\widehat{\mathcal{T}}} \exists S_i\}$. In particular, for every two concepts $\exists S_i, \exists S_j \in \mathfrak{C}$, if $\exists S_i \equiv_{\widehat{\mathcal{T}}} \exists S_j$ then $\exists S_i \in \mathfrak{C}(\exists S_j)$; but the implication on the other direction does not hold. Intuitively, if $\exists S_i, \exists S_j$ belong to the same cluster, then their extensions have the same cardinality in every finite model of $\widehat{\mathcal{T}}$, and of \mathcal{T} .

Further, we use $[\exists S_i] \succ [\exists S_j]$ to denote the fact that there exist concepts $\exists R \in [\exists S_i]$ and $\exists R' \in [\exists S_j]$ such that $\exists R \geq_{\widehat{\mathcal{T}}} \exists R'$, but $\exists R' \not\geq_{\widehat{\mathcal{T}}} \exists R$, i.e., $\exists R' \notin \mathfrak{C}(\exists R)$. For example, in the TBox \mathcal{T} from Example 2, $[\exists P_1^-] \succ [\exists S]$. Notably, $\exists P_1^- \geq_{\widehat{\mathcal{T}}} \exists S$, but $\exists S \notin \mathfrak{C}(\exists P_1^-) = \{\exists P_1, \exists P_1^-, \exists P_2, \exists P_2^-\}$. It is also the case that $[\exists P_1^-] \not\succeq [\exists P_2]$, since $\mathfrak{C}(\exists P_1^-) = \mathfrak{C}(\exists P_2)$.

For constructing the domain of the desired finite model $\mathcal{I}_{\mathcal{T}}^f$, we define the set of *finite paths* of $\mathcal{I}_{\widehat{\mathcal{T}}}$. $\sigma = (d_{S_0} \cdots d_{S_k}) \in \text{finpaths}(\mathcal{I}_{\widehat{\mathcal{T}}})$ iff σ satisfies the following conditions:

1. $[\exists S_i] \xrightarrow{R} [\exists S']$, for some role R , such that :
 - (a) $(\text{funct } R^-) \in \widehat{\mathcal{T}}$,
 - (b) $[\exists S_{i+1}] \in \mathfrak{C}(\exists S')$, and

- (c) $\exists R \notin [\exists S_{i+1}]$
- 2. $[\exists S_{i+1}] \succ [\exists S_i]$

Intuitively, by condition (1a) a path $(\sigma \cdot d_R^-)$ can be ‘reused’ as a witness of an existential $\exists R$, whenever the inverse of R is not functional, otherwise a new object $(\sigma' \cdot d_R^-)$ is needed as a witness. Condition (1b) ensures that whenever such a witness path $(\sigma \cdot d_{S'})$ belongs to $\text{finpaths}(\mathcal{I}_{\hat{\mathcal{T}}})$, then also witnesses $(\sigma \cdot d_{R_{i+1}})$ for each class $[\exists R_{i+1}]$ in the cluster $\mathfrak{C}(\exists S')$ belong to $\text{finpaths}(\mathcal{I}_{\hat{\mathcal{T}}})$; condition (1c) avoids to include a witness that is already realized by $\text{tail}(\sigma)$. Moreover, by condition 2 the length of every path is bounded, and since \mathfrak{C} is finite, then $\text{finpaths}(\mathcal{I}_{\hat{\mathcal{T}}})$ is also finite.

We consider a subset of $\text{finpaths}(\mathcal{I}_{\hat{\mathcal{T}}})$ as the domain of $\mathcal{I}_{\mathcal{T}}^f$ that it is determined by $\varphi = C \sqsubseteq B$. More specifically, for $\sigma = d_S \cdot \sigma' \in \text{finpaths}(\mathcal{I}_{\hat{\mathcal{T}}})$, we write $\varphi \rightsquigarrow \sigma$ iff there is a sequence of roles R_0, \dots, R_n such that:

1. $C \sqcap \neg B \sqsubseteq \exists R_0, \exists S \in [\exists R_n^-]$;
2. for $i \leq n$, $\exists R_i \in [\exists S_i]$, $[\exists S_i] \xrightarrow{R_i} [\exists S_{i+1}]$, and $\exists R_i \notin [\exists S_{i+1}]$;
3. either $(\text{funct } R_i^-) \notin \hat{\mathcal{T}}$ or $[\exists S_{i+1}] \not\prec [\exists S_i]$.

We are ready now to define $\mathcal{I}_{\mathcal{T}}^f$. We set $\Delta^{\mathcal{I}_{\mathcal{T}}^f} = \{\sigma \in \text{finpaths}(\mathcal{I}_{\hat{\mathcal{T}}}) \mid \varphi \rightsquigarrow \sigma\}$. For each concept name A , $A^{\mathcal{I}_{\mathcal{T}}^f} \subseteq \Delta^{\mathcal{I}_{\mathcal{T}}^f}$, and for each atomic role P , $P \subseteq (\Delta^{\mathcal{I}_{\mathcal{T}}^f} \times \Delta^{\mathcal{I}_{\mathcal{T}}^f})$, such that:

$$\begin{aligned}
A^{\mathcal{I}_{\mathcal{T}}^f} &= \{\sigma \in \Delta^{\mathcal{I}_{\mathcal{T}}^f} \mid \text{tail}(\sigma) \sqsubseteq_{\hat{\mathcal{T}}} A\}; \\
P_k^{\mathcal{I}_{\mathcal{T}}^f} &= \{((\sigma \cdot d_{S_i}), (\sigma \cdot d_{S_i} d_{S_j})) \mid [\exists S_i] \xrightarrow{P} [\exists S_j]\} \\
&\quad \cup \{((\sigma \cdot d_{S_i} d_{S_j}), (\sigma \cdot d_{S_i})) \mid [\exists S_j] \xrightarrow{P^-} [\exists S_i]\} \\
&\quad \cup \{(d_\varphi, (d_P^- \cdot \sigma)) \mid \varphi \sqsubseteq \exists P\} \\
&\quad \cup \{((d_P^- \cdot \sigma), d_\varphi) \mid \varphi \sqsubseteq \exists P^-\} \\
&\quad \cup \{((\sigma \cdot d_P), (\sigma', d_P^-)) \mid \sigma \neq \sigma', (\text{funct } P^-) \notin \hat{\mathcal{T}}\}.
\end{aligned}$$

As an example, consider the model $\mathcal{I}_{\mathcal{T}}^f$, shown in Figure 6 for the TBox \mathcal{T} from Example 2.

We claim that $\mathcal{I}_{\mathcal{T}}^f$ and $\mathcal{U}_{\hat{\mathcal{T}}}$ are g -equisimilar. Indeed, a global g -equisimulation ρ can be defined by $(\sigma, \gamma) \in \rho$ iff $\text{tail}(\gamma) = d_R$, $\text{tail}(\sigma) = d_S$ and $\exists R \in [\exists S]$.

Since $\mathcal{U}_{\hat{\mathcal{T}}} \models \hat{\mathcal{T}}$, by Lemma 1(b), $\mathcal{I}_{\mathcal{T}}^f \models \hat{\mathcal{T}}$; and since $\mathcal{T} \subseteq \hat{\mathcal{T}}$, $\mathcal{I}_{\mathcal{T}}^f \models \mathcal{T}$. Moreover, $\mathcal{I}_{\mathcal{T}}^f$ is as desired: $\mathcal{I}_{\mathcal{T}}^f \not\models C \sqsubseteq B$, since by construction $t_{\mathcal{I}_{\mathcal{T}}^f}(d_\varphi) = t_{\mathcal{U}_{\hat{\mathcal{T}}}}(d_\varphi)$. Finally, the case for $\varphi = (\text{funct } R)$, can be handled by a slight modification of the previous construction. Essentially, we substitute d_φ in the previous construction by a witness d_R with two R -successors.

From Lemma 3 and Lemma 4 we conclude that finite model TBox reasoning in $DL\text{-Lite}_{\text{horn}}^{\mathcal{F}}$ can be reduced to arbitrary TBox reasoning.

Theorem 1. *For a given $DL\text{-Lite}_{\text{horn}}^{\mathcal{F}}$ TBox \mathcal{T} , concepts C_1 and C_2 . We have that the following hold:*

1. \mathcal{T} is finitely satisfiable iff $\text{finClosure}(\mathcal{T})$ is satisfiable.
2. $\mathcal{T} \models_{\text{fin}} C_1 \sqsubseteq C_2$ iff $\text{finClosure}(\mathcal{T}) \models C_1 \sqsubseteq C_2$.

Next, we show that the complexity of finite model reasoning in $DL-Lite_{horn}^{\mathcal{F}}$ remains in PTIME, when considering also an ABox, i.e., the following hold:

Theorem 2. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $DL-Lite_{horn}^{\mathcal{F}}$ KB. Then \mathcal{K} is finitely satisfiable iff $(\text{finClosure}(\mathcal{T}), \mathcal{A})$ is satisfiable.*

Thus, the complexity of reasoning in $DL-Lite_{horn}^{\mathcal{F}}$ coincides for finite and arbitrary models.

Theorem 3. *Finite model reasoning in $DL-Lite_{horn}^{\mathcal{F}}$ is PTIME-complete.*

As pointed out in Section 3 the canonical interpretation of a knowledge base \mathcal{K} constructed as in Definition 1 it is not in general a model of \mathcal{K} , due to the presence of functionality constraints (and arbitrary number restrictions in general). The latter observation provides an intuition for the construction of $\mathcal{I}_{\hat{\mathcal{T}}}^f$. Intuitively, $\mathcal{I}_{\hat{\mathcal{T}}}$ can be transformed into a finite model by creating ‘copies’ of certain portions (clusters) in order to resolve violations to functionality constraints; then, although the number of R -successors of some objects in the model increases (specifically for those roles R in the TBox such that $(\text{funct } R) \notin \hat{\mathcal{T}}$), this does not trigger any inconsistency, because the expressive power of $DL-Lite_{horn}^{\mathcal{F}}$ allows only to distinguish between two types of objects: those with exactly one R -successor, and those with one or more. As we shall see on the next section this approach for constructing a finite model fails when considering arbitrary number restrictions.

5 Finite Model Reasoning in $DL-Lite_{horn}^{\mathcal{N}}$

Kontchakow et. al., [9] show the following result by a reduction of the SAT problem to finite satisfiability in $DL-Lite_{horn}^{\mathcal{N}}$.

Lemma 5 ([9, Remark 98]). *Finite satisfiability of $DL-Lite_{horn}^{\mathcal{N}}$ TBoxes is NP-hard.*

From the proof of the previous lemma, it can be seen that, contrary to the arbitrary model case, when restricting to finite models in $DL-Lite_{horn}^{\mathcal{N}}$, it is possible to express disjunctive knowledge, such as covering of a concept C by a disjunction of concepts, even though the disjunction operator, ‘ \sqcup ’, is not part of the logic. Moreover, $DL-Lite_{horn}^{\mathcal{N}}$ loses convexity when restricting to finite models. In order to understand this more clearly, let us consider a TBox \mathcal{T} with the following axioms:

$$\begin{aligned} &\geq 3 P_1 \sqsubseteq \perp, \quad \exists P_1 \sqsubseteq \geq 2 P_1, \quad \geq 2 P_1 \equiv \exists P_2, \quad \top \sqsubseteq \exists P_1^-, \quad (\text{funct } P_1^-), \\ &B_2 \equiv \exists P_2^-, \quad (\text{funct } P_2), \quad (\text{funct } P_2^-), \quad B_1 \sqcap B_2 \sqsubseteq \perp. \end{aligned}$$

Let \mathcal{I} be a finite model of \mathcal{T} , and $N = \sharp(\Delta^{\mathcal{I}})$. We have that $N = 2 \cdot \sharp(\geq 2 P_1)^{\mathcal{I}}$. Furthermore, $\sharp(\geq 2 P_1) = \sharp(B_2)^{\mathcal{I}} = M$, since $P_2^{\mathcal{I}}$ is a bijective function; and since B_1 is disjoint with B_2 , then $\sharp(B_1)^{\mathcal{I}} = M$. Hence, $\Delta^{\mathcal{I}} = (B_1)^{\mathcal{I}} \cup (B_2)^{\mathcal{I}}$; and as a consequence $\mathcal{T} \models_{\text{fin}} \geq 2 P_1 \sqsubseteq B_1 \sqcup B_2$. However, $\mathcal{T} \not\models \geq 2 P_1 \sqsubseteq B_1$, and $\mathcal{T} \not\models \geq 2 P_1 \sqsubseteq B_2$.

The best known upper bound for finite satisfiability in $DL\text{-}Lite_{horn}^N$ is EXP-TIME, which is given by the complexity of finite model reasoning in \mathcal{ALCQI} [11]. The approach taken by Lutz et. al., [11] is to transform a given \mathcal{ALCQI} TBox into a system of linear inequalities which is exponential on the size of the TBox. We conjecture that this exponential blow up can be avoided when considering $DL\text{-}Lite_{horn}^N$ TBoxes. The combinatorial nature of this problem suggests indeed the use of techniques of linear programming. However, we consider that a reduction of this problem to the fragment of FOL with one variable and counting quantifiers is also feasible. For devising ad hoc algorithms for finite model reasoning in DLs, it is still relevant to propose a constructive approach as in the case of $DL\text{-}Lite_{horn}^F$. All these research problems, as well as constructions of finite models of KBs in logics in the \mathcal{EL} family constitute ongoing research.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyashev, M.: Reasoning over extended ER models. In: Proc. of the 26th Int. Conf. on Conceptual Modeling (ER 2007). Lecture Notes in Computer Science, vol. 4801, pp. 277–292. Springer (2007)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: The $DL\text{-}Lite$ family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope further. In: Clark, K., Patel-Schneider, P.F. (eds.) Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008) (2008)
4. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence 168(1–2), 70–118 (2005)
5. Borgida, A., Brachman, R.J.: The description logic handbook: theory, implementation, and applications, chap. Conceptual Modeling with Description Logics, pp. 349–372. Cambridge University Press (2003)
6. Cosmadakis, S.S., Kanellakis, P.C., Vardi, M.Y.: Polynomial-time implication problems for unary inclusion dependencies. J. ACM 37(1), 15–46 (1990)
7. Goranko, V., Otto, M.: Handbook of Modal Logic, chap. Model Theory of Modal Logic, pp. 255–325. Elsevier (2006)
8. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to query answering in $DL\text{-}Lite$. In: Lin, F., Sattler, U. (eds.) Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010). AAAI Press (2010)
9. Kontchakov, R., Wolter, F., Zakharyashev, M.: Logic-based ontology comparison and module extraction with an application to $DL\text{-}Lite$. AIJ 174(15), 1093–1141 (2010)
10. Lutz, C., Piro, R., Wolter, F.: Description logic TBoxes: Model-theoretic characterizations and rewritability. In: Proc. of the 22st Int. Joint Conf. on Artificial Intelligence (IJCAI 2012). AAAI Press (2011)
11. Lutz, C., Sattler, U., Tendra, L.: The complexity of finite model reasoning in description logics. Information and Computation 199, 132–171 (2005)
12. de Rijke, M.: A note on graded modal logic. Studia Logica 64(2), 271–283 (2000)
13. Rosati, R.: Finite model reasoning in $DL\text{-}Lite$. In: Proc. of the 5th Eur. Semantic Web Conf. (ESWC 2008). Lecture Notes in Computer Science, vol. 5021, pp. 215–229 (2008)

An Abstract Tableau Calculus for the Description Logic \mathcal{SHOI} Using Unrestricted Blocking and Rewriting

Mohammad Khodadadi, Renate A. Schmidt, and Dmitry Tishkovsky*

School of Computer Science, The University of Manchester, UK

Abstract This paper presents an abstract tableau calculus for the description logic \mathcal{SHOI} . \mathcal{SHOI} is the extension of \mathcal{ALC} with singleton concepts, role inverse, transitive roles and role inclusion axioms. The presented tableau calculus is inspired by a recently introduced tableau synthesis framework. Termination is achieved by a variation of the unrestricted blocking mechanism that immediately rewrites terms with respect to the conjectured equalities. This approach leads to reduced search space for decision procedures based on the calculus. We also discuss restrictions of the application of the blocking rule by means of additional side conditions and/or additional premises.

1 Introduction

Since the late nineteen eighties various tableau algorithms have been developed for description logics [2]. The way they are defined and blocking is performed these tableau algorithms exploit in an essential way that the supported description logics have a kind of tree-model property. The basic idea is to perform the derivations such that tree-like models are constructed by systematically creating maximally expanded label sets of concept expressions for individual terms one-by-one in a stratified way. Blocking can then be used to ensure no two individuals (perhaps, in an ancestor relationship) have the same label sets, or are a subset of other label sets. For description logics with role inverse, nominals and number restrictions, this kind of stratified construction is more complex requiring some back-and-forth traversal of a tree model together with forms of dynamic blocking [9,10]. This more complex non-local construction is still aimed at finding tree models and therefore not sufficient for description logics without a kind of tree-model property.

In [14,13] we show description logics without the tree-model property, in particular, the description logics \mathcal{ALBO} and $\mathcal{ALBO}^{\text{id}}$, can be decided using a labelled tableau approach enhanced with the so-called unrestricted blocking mechanism. Labelled tableau approaches are common for modal logics, hybrid logics and various other non-classical logics, cf. e.g., [5,3,4,1]. Labelled tableau approaches are easy to understand, they are easy to define as abstract calculi,

* This research is supported by EPSRC research grant EP/H043748/1.

even for undecidable logics, and are not limited to logics with a form of tree model property. There is also more flexibility in the way that derivations can be performed and it is thus easy to devise sound and complete tableau calculi. Building on [14,13] we have devised a framework for systematically developing labelled tableau calculi for various logics, not only description logics [12]. Essentially for any logic whose semantic definition can be specified in the specification language of the framework, a sound and complete labelled tableau calculus can be synthesised, if certain general conditions hold.

Being based on a sound tableau rule and equality reasoning, the unrestricted blocking mechanism is generally sound and can be incorporated into sound and complete labelled tableau calculi or related approaches. We have shown that if the logic has the finite model property then adding the unrestricted blocking mechanism guarantees also termination [11,12]. Unrestricted blocking provides an intuitively simple method for obtaining termination and behaves very different to standard blocking techniques. It does not require specialised blocking tests and complicated dynamic processing steps. All individuals are blockable and once blocked remain blocked. It can be used to find small finite models.

The aim of this paper is to formalise reasoning for a well-studied, expressive description logic in an abstract labelled tableau calculus incorporating unrestricted blocking. We also want to explore the possibilities of emulating different kinds of existing blocking techniques. In particular, we present an abstract labelled tableau calculus for the description logic *SHOI*. The tableau calculus is in line with a refined tableau calculus obtained in the tableau synthesis framework, but exploiting the tree model property of *SHOI*, transitive roles are accommodated via a propagation rule rather than a structural rule.

Different to most labelled tableau approaches the expansion of \exists expressions introduces Skolem terms rather than constants. Another novelty is the use of ordered rewriting to realise equality reasoning for singleton concepts (nominals) and blocking. Though there are similarities with substitution and nominal deletion approaches (e.g., [10,4,1]), using Skolem terms and ordered rewriting avoids the need to perform again some inference steps on the same branch. Also significantly fewer inferences are performed than when using standard tableau rules for equality as in, e.g., [3,14,13]. As the unrestricted blocking rule is generally sound, any restriction of the rule obtained by adding side-conditions or premises is also sound. This makes it possible to restrict the application of blocking without losing soundness and completeness. For example, it is possible to approximate standard loop checking techniques such as subset ancestor blocking or anywhere equality blocking and simulate approaches using the δ^* -rule.

The paper is structured as follows. The syntax and semantics of *SHOI* are defined in Section 2. In Section 3 we define the tableau calculus for *SHOI* and in Section 4 we prove that it is sound, complete and terminating. Furthermore, we present examples of restricting the blocking rule by imposing constraints via additional premises and/or side conditions in Section 4. Due to space restrictions we do not discuss the emulation of all known existing blocking techniques but the examples given illustrate the general idea.

2 The Description Logic \mathcal{SHOI}

\mathcal{SHOI} extends the description logic \mathcal{ALC} with singleton concepts, role inverse, transitive roles and role inclusion axioms. The language of \mathcal{SHOI} is defined over disjoint countable sets of concept names (atomic concepts), individuals, and role names (atomic roles). The symbol A is used to denote an atomic concept, the symbols a and b denote individuals, and the symbol r denotes an atomic role. Concept and role expressions are built from atomic concepts, individuals, and atomic roles using connectives $\{\cdot\}$ (singleton operator), \neg , \sqcup , and $\exists \cdot \cdot$ (existential restriction operator), $^-$ (role inverse operator). Formally, concept and role expressions are defined respectively by the following grammar rules, where C and D denote concept expressions and R denotes a role expression.

$$\begin{aligned} C, D &\stackrel{\text{def}}{=} A \mid \{a\} \mid \neg C \mid C \sqcup D \mid \exists R.C \\ R &\stackrel{\text{def}}{=} r \mid R^- \end{aligned}$$

The operators \top , \perp , \sqcap , and $\forall \cdot \cdot$ are defined as usual. In order to simplify the syntax and avoid repetitive occurrences of the role inverse operator we assume that $(r^-)^- \stackrel{\text{def}}{=} r$. Further, in \mathcal{SHOI} , any atomic role is allowed to be declared as transitive and the predicate Trans is used to denote this. Thus, for every atomic role r , $\text{Trans}(r)$ is true iff r is transitive.

A description logic knowledge base consists of an ABox \mathcal{A} , a TBox \mathcal{T} and an RBox \mathcal{R} . The ABox consists of a finite number of concept assertions of the form $a : C$ and role assertions of the form $(a, b) : R$. The TBox is used to express a hierarchy between concepts through a finite set of inclusion statements of the form $C \sqsubseteq D$. A *normalised* TBox is a set of inclusion statements of the form $\top \sqsubseteq C$. The RBox is a finite set of inclusion statements of the form $R \sqsubseteq S$ and $\text{Trans}(r)$, to specify a hierarchy between roles and define transitivity of some roles. We define the *closure* \mathcal{R}^+ of the RBox \mathcal{R} as the smallest RBox that contains \mathcal{R} and satisfies the following properties.

- if $Q \sqsubseteq R \in \mathcal{R}^+$ then $Q^- \sqsubseteq R^- \in \mathcal{R}^+$;
- if $Q \sqsubseteq R, R \sqsubseteq S \in \mathcal{R}^+$ then $Q \sqsubseteq S \in \mathcal{R}^+$.

Given an RBox \mathcal{R} , let \mathcal{R}^* denote the RBox $\mathcal{R}^+ \cup \{R \sqsubseteq R \mid R \text{ is a role}\}$.

The semantics of \mathcal{SHOI} is defined by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ given by a pair of a non-empty set $\Delta^{\mathcal{I}}$, referred to as the *domain* of interpretation, and an interpretation function $\cdot^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ maps individuals to elements of the domain, concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. Regarding roles declared as being transitive, $\cdot^{\mathcal{I}}$ must satisfy that $r^{\mathcal{I}}$ is a transitive relation whenever $\text{Trans}(r)$ is true. The function $\cdot^{\mathcal{I}}$ extends to all concept and role expressions by induction on lengths of expressions as follows:

$$\begin{aligned} a^{\mathcal{I}} &\stackrel{\text{def}}{=} \{a^{\mathcal{I}}\}, \quad (\neg C)^{\mathcal{I}} \stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \quad (C \sqcup D)^{\mathcal{I}} \stackrel{\text{def}}{=} C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\ (\exists R.C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \{x \mid \exists y \in C^{\mathcal{I}} (x, y) \in R^{\mathcal{I}}\}, \quad (R^-)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(x, y) \mid (y, x) \in R^{\mathcal{I}}\}. \end{aligned}$$

According to the semantics, the inverse of a role is transitive iff the role is transitive. Following this, we extend the predicate Trans to all role expressions so that $\text{Trans}(r^-)$ is true iff $\text{Trans}(r)$ is true.

Let E denote any concept expression, any concept inclusion, any role inclusion, any concept assertion or any role assertion. We indicate by $\mathcal{I} \models E$ that E is *valid* in the model \mathcal{I} . We define:

$$\begin{array}{ll} \mathcal{I} \models C & \stackrel{\text{def}}{\iff} C^{\mathcal{I}} = \Delta^{\mathcal{I}} & \mathcal{I} \models a : C & \stackrel{\text{def}}{\iff} a^{\mathcal{I}} \in C^{\mathcal{I}} \\ \mathcal{I} \models R \sqsubseteq S & \stackrel{\text{def}}{\iff} R^{\mathcal{I}} \subseteq S^{\mathcal{I}} & \mathcal{I} \models (a, b) : R & \stackrel{\text{def}}{\iff} (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}} \\ \mathcal{I} \models C \sqsubseteq D & \stackrel{\text{def}}{\iff} C^{\mathcal{I}} \subseteq D^{\mathcal{I}} & & \end{array}$$

Because \mathcal{SHOI} supports singleton concepts, every ABox statement $a : C$ can be encoded by the TBox statement $\{a\} \sqsubseteq C$. Also, every role assertion $(a, b) : R$ can be encoded as the TBox statement $\{a\} \sqsubseteq \exists R.\{b\}$. Thus, without loss of generality, we assume that a knowledge base is a pair $(\mathcal{T}, \mathcal{R})$ which consists of a normalised TBox \mathcal{T} and an RBox \mathcal{R} . It worth noting that the TBox can be internalised as well [15] but for performance reasons we present a tableau calculus that handles TBox statements directly.

A concept C is *satisfiable* in a model \mathcal{I} iff $C^{\mathcal{I}} \neq \emptyset$. A concept is *satisfiable in \mathcal{I} with respect to a knowledge base* if it is satisfiable in \mathcal{I} whenever every statement of the knowledge base is valid in \mathcal{I} . That is, C is satisfiable with respect to $(\mathcal{T}, \mathcal{R})$ in \mathcal{I} iff $C^{\mathcal{I}} \neq \emptyset$ provided that $\mathcal{I} \models E$ for every $E \in \mathcal{T} \cup \mathcal{R}$.

3 An Abstract Tableau Calculus for \mathcal{SHOI}

In this section we present a labelled semantic ground tableau calculus for \mathcal{SHOI} .

The language of the tableau calculus is an extension of the language of \mathcal{SHOI} with equality formulae and individual terms used as labels. We add a function symbol f which takes a triple (s, R, C) consisting of an individual term s , a role expression R and a concept expression C as its arguments and define the set of (*individual*) *terms* s inductively by the following grammar rule, where a denotes any individual, C any concept and R any role.

$$s \stackrel{\text{def}}{=} a \mid f(s, R, C)$$

Terms which are not ABox individuals can be viewed as being Skolem terms.

Formulae in the tableau language are defined by the following grammar rule, where s and t are individual terms, C is a concept and R is a role.

$$E \stackrel{\text{def}}{=} s : C \mid (s, t) : R \mid s \approx t$$

We extend the interpretation of \mathcal{SHOI} expressions to the formulae of the tableau language. For every \mathcal{SHOI} interpretation \mathcal{I} , let the interpretation $f^{\mathcal{I}}$ in \mathcal{I} of the function f be an arbitrary function mapping triples (x, ρ, χ) with $x \in \Delta^{\mathcal{I}}$, $\rho \subseteq (\Delta^{\mathcal{I}})^2$, $\chi \subseteq \Delta^{\mathcal{I}}$ to elements of $\Delta^{\mathcal{I}}$. We let

$$\begin{array}{ll} (f(a, R, C))^{\mathcal{I}} & \stackrel{\text{def}}{=} f^{\mathcal{I}}(a^{\mathcal{I}}, R^{\mathcal{I}}, C^{\mathcal{I}}), & \mathcal{I} \models (s : C)^{\mathcal{I}} & \stackrel{\text{def}}{\iff} s^{\mathcal{I}} \in C^{\mathcal{I}}, \\ \mathcal{I} \models s \approx t & \stackrel{\text{def}}{\iff} s^{\mathcal{I}} = t^{\mathcal{I}}, & \mathcal{I} \models (s, t) : R & \stackrel{\text{def}}{\iff} (s^{\mathcal{I}}, t^{\mathcal{I}}) \in R^{\mathcal{I}}. \end{array}$$

The interpretations of the formulae $s \approx t$, $s : \{t\}$ and $t : \{s\}$ coincide. In accordance with their interpretation we refer to these formulae as *equalities*. We also refer to the formulae of the form $s : \neg\{t\}$ as *inequalities*.

Let Tab denote a tableau calculus comprising of a set of inference rules. A *derivation* or *tableau* for Tab is a finitely branching, ordered tree whose nodes are annotated by sets of tableau formulae. Assuming that C is the input concept expression to be tested for satisfiability with respect a knowledge base $(\mathcal{T}, \mathcal{R})$ the root node of the tableau is the set $\{a : C\}$, where a denotes a fresh individual. Successor nodes are constructed in accordance with a set of inference rules in the calculus. The inference rules have the general form

$$\frac{X_0}{X_1 \mid \dots \mid X_n} \text{ (side-condition),}$$

where X_0 is the set of premises and the X_i are the sets of conclusions. If $n = 0$, the rule is called *closure rule* and written X_0/\perp .

If σ is a substitution that acts on tableau formulae and $X = \{E_1, \dots, E_k\}$ is a set of tableau formulae then $X\sigma$ denotes the set $\{E_1\sigma, \dots, E_k\sigma\}$. A rule is *applicable* to a tableau if there is a leaf node annotated with a set N and there is a substitution σ such that $X_0\sigma \subseteq N$, where X_0 is the set of premises of the rule, and the side-condition of the rule is true for N . σ is called the *matching substitution* of the rule application. We assume in a rule individual symbols, concept symbols and role symbols represent variables that are matched with individual terms, concept expressions and role expressions respectively. We also say the rule is applicable to the formulae $X_0\sigma$ in (the leaf node of) the branch.

If a rule of the calculus is applicable to a leaf node of the tableau with a matching substitution σ , then the tableau is extended by attaching to the leaf node n child nodes annotated with $N \cup X_i\sigma$ for $i = 1, \dots, n$, respectively. In order to avoid redundancies we stipulate that a rule application to a leaf node annotated with N is *redundant* if there is a conclusion set X_i for some $i = 1, \dots, n$ of the rule such that $X_i\sigma \subseteq N$, where σ is the matching substitution. This ensures rules are not applied more than once to the same sets of formulae.

A *branch* in the tableau is a maximal path from the root of the tableau to a leaf node. If a closure rule has been applied in a branch then the branch is said to be *closed*. If a branch is not closed, it is called *open*. A tableau is *closed* if all its branches are closed. A branch is *fully expanded* if no more rules are applicable to its leaf node modulo redundancy. We call a tableau *fully expanded* iff all its branches are fully expanded. We denote by $Tab(\mathcal{T}, \mathcal{R}, C)$ a fully expanded tableau constructed in the calculus Tab for the input concept C and the knowledge base $(\mathcal{T}, \mathcal{R})$.

We need equality reasoning for individual terms to achieve termination for the calculus. Equality reasoning can be provided in various ways. One is to supply special tableau rules for reasoning modulo equalities within the branch in a similar way as is done in [3,14,13]. Another is to use ordered term rewriting. Ordered rewriting is more efficient for handling equal individuals because it allows to reduce the number of tableau formulae in the current branch. Since

$$\begin{array}{l}
(\perp): \frac{s : \neg C, s : C}{\perp} \\
(\sqcup): \frac{s : C \sqcup D}{s : C \mid s : D} \\
(\exists): \frac{s : \exists R.C}{f(s, R, C) : C, (s, f(s, R, C)) : R} \\
(\neg\exists): \frac{s : \neg\exists S.C, (s, t) : R}{t : \neg C} \quad (R \sqsubseteq S \in \mathcal{R}^*) \\
(\neg\exists^-): \frac{s : \neg\exists S^-.C, (t, s) : R}{t : \neg C} \quad (R \sqsubseteq S \in \mathcal{R}^*) \\
(+): \frac{s : \neg\exists S.C, (s, t) : R}{t : \neg\exists R.C} \quad (R \sqsubseteq S \in \mathcal{R}^*, \text{Trans}(R) \in \mathcal{R}) \\
(-^+): \frac{s : \neg\exists S^-.C, (t, s) : R}{t : \neg\exists R^-.C} \quad (R \sqsubseteq S \in \mathcal{R}^*, \text{Trans}(R) \in \mathcal{R}) \\
(\text{RBox}): \frac{(s, t) : R}{(s, t) : S} \quad (R \sqsubseteq S \in \mathcal{R}^+) \\
(\neg\neg): \frac{s : \neg\neg C}{s : C} \\
(\neg\sqcup): \frac{s : \neg(C \sqcup D)}{s : \neg C, s : \neg D} \\
(-): \frac{(s, t) : R^-}{(t, s) : R} \\
(\text{id}): \frac{s : C}{s : \{s\}} \\
(\text{id}_2): \frac{s : \neg\{t\}}{t : \{t\}} \\
(\text{cng}): \frac{(s, t) : R}{s : \{s\}, t : \{t\}} \\
(\text{TBox}): \frac{s : \{s\}}{s : C} \quad (C \in \mathcal{T}) \\
(\approx): \frac{s : \{t\}}{s \approx t} \quad (s \neq t)
\end{array}$$

Figure 1. The tableau calculus $\text{Tab}_{\mathcal{SHOI}}$

all individual terms in any tableau derivation are ground we are dealing with a special case of rewriting, namely, ground rewriting.

In this paper, a *rewrite system* R is a binary relation on the set of all individual terms and consists of rewrite rules which are pairs of individual terms. In order to handle equalities, we orient each equality formula appearing in the current branch of a tableau derivation according to a special ordering \succ which is a strict partial order on individual terms. We denote by $s \rightarrow t$ a rewrite rule (s, t) in which $s \succ t$. Thus, if an equality formula $s \approx t$ appears in a node of a branch then either $s \rightarrow t$ or $t \rightarrow s$ is added as a rewrite rule to the rewrite system of the branch. A term which cannot be rewritten (with respect to a rewrite system) is said to be in *normal form*. A normal form of a term s is denoted by $\text{nf}(s)$. A rewrite system is *terminating* if there is a normal form for each term.

Our tableau calculus $\text{Tab}_{\mathcal{SHOI}}$ for the description logic \mathcal{SHOI} is given in Figure 1. The (\perp) rule is the closure rule. The $(\neg\neg)$ rule removes occurrences of double negation on concepts. The (\sqcup) and $(\neg\sqcup)$ rules are standard rules for handling concept disjunctions. Given a tableau formula $s : \exists R.C$, the (\exists) rule introduces an individual term $f(s, R, C)$ as an R -successor of s (instead of introducing a fresh individual as might be done in other presentations). The $(\neg\exists)$ rule is equivalent to the standard rule for universally restricted concept expressions. The $(\neg\exists^-)$ rule allows the backward propagation of concept expressions along inverted links. The $(-)$ rule inverts a given link.

The $(+)$ rule propagates negated existential concept restriction along a transitive link while the $(-^+)$ rule does the same for inverse occurrences of transitive roles. Equalities of the form $s : \{s\}$ are tautologies, used in our calculus as domain predicates for keeping track of the terms that have been introduced to a branch. This is achieved with the three rules (id) , (id_2) and (cng) . The (TBox) rule concatenates every concept of the normalised TBox with every label occur-

ring on the branch. The (RBox) propagates a link of a role into its superrole according to the closure \mathcal{R}^+ of the given RBox \mathcal{R} .

The (\approx) rule is a special rule adding, what we call, a *rewrite trigger* $s \approx t$ to the branch. Let \succ be any reduction ordering on the set of individuals in the branch. The addition of any tableau formula $s \approx t$ to a set N of formulae which annotates a leaf tableau node immediately triggers the following rewrite process. Suppose that $s \succ t$ (the case $t \succ s$ is symmetrical). Then, $s \rightarrow t$ is added to a rewrite system R associated with the current tableau branch. The tableau is extended by attaching one child node to the current leaf node. The child node is annotated by the set N' obtained by rewriting all the tableau formulae in N with respect to the rewrite system R . In particular, this means that, in N' every term s is replaced by a term u such that $s \xrightarrow{*} u$ with respect to R .

4 Soundness, Completeness and Termination

It is not difficult to see that each rule of Tab_{SHOI} is sound, i.e., preserves satisfiability of concept assertions. Consequently:

Theorem 1 (Soundness). *The tableau calculus Tab_{SHOI} is sound for the description logic $SHOI$. That is, if a concept C is satisfiable with respect to the knowledge base $(\mathcal{T}, \mathcal{R})$ then any fully expanded Tab_{SHOI} -tableau for $(\mathcal{T}, \mathcal{R}, C)$ has an open branch.*

A tableau calculus Tab is *complete* iff for every knowledge base $(\mathcal{T}, \mathcal{R})$ and every concept C if C is unsatisfiable with respect to $(\mathcal{T}, \mathcal{R})$ then there is a closed tableau $Tab(\mathcal{T}, \mathcal{R}, C)$. In order to prove completeness of Tab_{SHOI} , we prove its constructive completeness which implies completeness. A tableau calculus Tab is *constructively complete* if for every open branch in any fully expanded tableau $Tab(\mathcal{T}, \mathcal{R}, C)$ there is a model which validates the knowledge base $(\mathcal{T}, \mathcal{R})$ and satisfies C .

Theorem 2 (Completeness). *Tab_{SHOI} is a (constructively) complete tableau calculus for the description logic $SHOI$.*

Next, we establish termination. A tableau calculus Tab is (*weakly*) *terminating* if any tableau $Tab(\mathcal{T}, \mathcal{R}, C)$ has a finite open branch provided that C is satisfiable concept with respect to the knowledge base $(\mathcal{T}, \mathcal{R})$.

Although Tab_{SHOI} is a sound and complete tableau calculus for the description logic $SHOI$, it is not terminating. In order to achieve termination, a form of blocking or loop-checking is necessary. One possibility is to add the *unrestricted blocking mechanism* described in [14]. As is shown in [12], this will ensure termination of an arbitrary tableau calculus under certain conditions, one of which being condition (c2) discussed below.

In this paper, we take a slightly different route and introduce a modified version of the unrestricted blocking mechanism. It is given by the (ub-rw) rule and ordered rewriting.

$$\text{(ub-rw): } \frac{s : \{s\}, t : \{t\}}{s \approx t \mid s : \neg\{t\}} \quad (s \neq t)$$

Here, $s \approx t$ is a rewrite trigger as introduced in Section 3. Premises of this rule are instantiated with any two distinct terms s and t used as labels in a set of tableau formulae N annotating the current leaf node. As a result of a rule application two successor nodes are created. If $s \succ t$ (respectively $t \succ s$) then in the left node a rewrite rule $s \rightarrow t$ (respectively $t \rightarrow s$) is added to the rewrite system \mathcal{R} . The left node is annotated with a copy of N , which is rewritten with respect to the newly obtained rewrite system \mathcal{R} . The right node is annotated with a copy of N extended with the additional formula $s : \neg\{t\}$. This formula indicates the case that s and t are not equal.

The calculus consisting of all the rules of $Tab_{\mathcal{SHOI}}$ and the rule (ub-rw) is denoted by $Tab_{\mathcal{SHOI}}(\text{ub-rw})$. Clearly, the (ub-rw) rule is sound. Therefore:

Theorem 3. *The calculus $Tab_{\mathcal{SHOI}}(\text{ub-rw})$ is a sound and (constructively) complete for the description logic \mathcal{SHOI} .*

In order to ensure termination for a procedure based on $Tab_{\mathcal{SHOI}}(\text{ub-rw})$ the rule application strategy must satisfy the following condition.

- (c2) In every open branch there is some node from which point onward before any application of the (\exists) rule all possible applications of the (ub-rw) rule have been performed.

(The unrestricted blocking mechanism in [14] also needs to satisfy a second condition, which is already satisfied in our modified setting.)

Provided that condition (c2) holds, a sufficient and necessary condition for termination of the tableau procedures based on $Tab_{\mathcal{SHOI}}(\text{ub-rw})$ is that \mathcal{SHOI} has the finite model property with respect to its standard semantics. This can be shown in a similar way as in [13]. A description logic has the *finite model property* if for an arbitrary concept C and arbitrary knowledge base it holds that if C is satisfiable with respect to the knowledge base in a model for the logic then C is satisfiable with respect to the knowledge base in a *finite* model of the logic.

The finite model property for \mathcal{SHOI} can be shown by a filtration argument.

Theorem 4 (Finite model property of \mathcal{SHOI}). *The description logic \mathcal{SHOI} has the finite model property.*

Therefore, using the results of [13] we obtain the following theorem.

Theorem 5 (Termination). *Any implementation, fair in the sense of [13], of the tableau calculus $Tab_{\mathcal{SHOI}}(\text{ub-rw})$ and satisfies condition (c2) is a decision procedure for \mathcal{SHOI} and its sublogics.*

5 Sound Restricted Blocking

The (ub-rw) rule creates potentially many branching points in a derivation, especially if the number individuals and \exists -expressions in the knowledge base is high. A way to reduce the number of applications of the (ub-rw) rule and thus

reduce the search space is to apply the blocking rule less often. This can be achieved by adding side-conditions and/or premises to the rule. Ideal would be side-conditions and additional premises that maximise the chance of constructing a finite model without the need for backtracking.

In the remaining section, we give some examples of restricted versions of the (ub-rw) rule. They all preserve soundness and completeness. We have:

Theorem 6 (Soundness and completeness). *The (ub-rw) rule constrained by additional premises or side-conditions is sound. Thus, $\text{Tab}_{\mathcal{SHOI}}$ extended with such a modified rule is sound and complete for \mathcal{SHOI} .*

Most existing description logic tableau algorithms aim to construct models given by relational tree structures where the nodes are individuals (or individual terms, if Skolem terms would have been used) and are annotated with label sets of concept expressions. A *label set* of a term s is the set $L(s) \stackrel{\text{def}}{=} \{C \mid s : C \in N\}$. These label sets are then used in the tests of standard blocking mechanism such as subset ancestor blocking and dynamic anywhere equality blocking.

An *emulation of subset ancestor blocking* can be realised through the selective application of the (ub-rw) rule, realised by adding a side-condition:

$$(\text{ub}_{\subseteq}\text{-rw}): \frac{s : \{s\}, t : \{t\}}{s \approx t \mid s : \neg\{t\}} \quad (s \neq t, s \text{ is an ancestor of } t \text{ and } L(t) \subseteq L(s))$$

In this rule the application of the (ub-rw) rule is restricted to a term s and its successor term t , where the label set of s is a superset of the label set of t . In our setting, as the calculus creates Skolem terms in the (\exists) rule, a term s is an ancestor of a term t , if s is a subterm of t .

Standard ancestor subset blocking is used in tableau algorithms for description logics \mathcal{ALC} , \mathcal{S} and \mathcal{SH} [7]. In ancestor subset blocking, a term t is blocked by its ancestor s if $L(t) \subseteq L(s)$. No rule is applicable to the blocked individuals. As standard ancestor blocking is not a branching rule it is important to perform the expansions in a stratified way and perform the subset test at an appropriate moment in order to preserve soundness. But even if the expansions are performed in the required way standard ancestor blocking is not generally sound unlike blocking based on the $(\text{ub}_{\subseteq}\text{-rw})$ rule.

Application of the (ub-rw) rule can be limited by ignoring the pairs of terms where the application of the rule is not critical for termination. E.g., it is possible to ignore pairs where both terms appear before some fixed node of a tableau derivation. We believe, as there are a finite number of individuals before a fixed tableau node, excluding them does not endanger termination. In particular, the pairs where both terms are ABox individuals can be ignored as in [8]. If the unique name assumption is assumed for the given ABox individuals, identifying these individuals by blocking would be incorrect. Using the following rule instead of using the (ub-rw) rule can have a significant impact on the performance, especially when reasoning over knowledge bases with a large number of individuals.

$$(\text{ub}_{\text{No ABox}\text{-rw}}): \frac{s : \{s\}, t : \{t\}}{s \approx t \mid s : \neg\{t\}} \quad (s \neq t, \text{ not both } s \text{ and } t \text{ are ABox individuals})$$

We can define a variation of the (ub-rw) rule restricted to the terms that are known to be the ones that may cause infinite derivations. For Tab_{SHOI} , infinite derivations can be caused only by infinite applications of the (\exists) rule. This means we may focus blocking on the terms to which the (\exists) rule may eventually be applicable, i.e., the terms which have an \exists -expression in their label sets. We may formulate the (ub-rw) rule as follows to reflect this restriction:

$$(\text{ub}_{\exists}\text{-rw}): \frac{s : \exists R.C, t : \exists S.D}{s \approx t \mid s : \neg\{t\}} \quad (s \neq t)$$

Here, $\exists R.C$, $\exists S.D$ are two \exists -expression that can be matched with any \exists -expression. This rule is applicable to any pair of terms s and t which both have a \exists -expression in their label sets.

The three variations of the (ub-rw) rule just presented are all sound, thus preserving soundness (and completeness) of the calculus is not an issue. An issue is to show under which conditions and for which logics termination can be ensured. Because of the side-conditions or additional premises these variations of the (ub-rw) rule are no longer applied to every possible pair of terms. Thus, condition (c2) does not hold. We believe however it can be proved that search strategies can be adopted where blocking applies to sufficiently many pairs so that the procedure terminates.

Next we illustrate how the (δ^*) rule [6] can be simulated using a restriction of the (ub-rw) rule. The (δ^*) rule systematically reuses terms in order to find finite models. For description logics the (δ^*) rule is defined as follows:

$$(\delta^*): \frac{s : \exists R.C}{(s, t_1) : R, t_1 : C \mid \dots \mid (s, t_n) : R, t_n : C \mid (s, f(s, R, C)) : R, f(s, R, C) : C}$$

Here, t_1, \dots, t_n are all existing terms, covering all given ABox individuals and all introduced Skolem terms on the current branch. The (δ^*) rule is actually a modified version of the (\exists) rule. Instead of creating a new term to satisfy an \exists -expression, this rule tries to satisfy the \exists -expression by reusing existing terms. If all the attempts to satisfy the \exists -expression with existing terms end in contradictions, then a new term $f(s, R, C)$ is introduced.

In our abstract calculus we can simulate the (δ^*) rule with the (\exists) rule and modifying the (ub-rw) rule to:

$$(\text{ub}_{\delta^*}\text{-rw}): \frac{s : \{s\}, t : \{t\}}{s \approx t \mid s : \neg\{t\}} \quad (s \neq t, t \text{ is a Skolem term})$$

We should use a rule application strategy where after each application of the (\exists) rule, the $(\text{ub}_{\delta^*}\text{-rw})$ rule is applied to the newly added Skolem term and every existing term. In contrast to the previous blocking variations, the $(\text{ub}_{\delta^*}\text{-rw})$ rule satisfies condition (c2), since all possible term comparisons are performed before any application of the (\exists) rule. Hence termination is ensured.

Theorem 7 (Termination). *Any implementation, fair in the sense of [13], of the tableau calculus Tab_{SHOI} extended with the $(\text{ub}_{\delta^*}\text{-rw})$ rule and using the described strategy is a decision procedure for $SHOI$ and its sublogics.*

6 Concluding Remarks

The contribution of this paper is an abstract labelled tableau calculus for the description logic *SHOI* using ordered rewriting and generic forms of blocking defined as variations of the unrestricted blocking mechanism. The tableau calculus is designed to be as general as possible in order to gain greater insight into minimal requirements for soundness, completeness and termination and conduct the proofs without any considerations for search strategies, heuristics and other implementation issues. The discussion in [13] of how to obtain deterministic tableau procedures for implementation based on the notion of fairness as defined in that paper carries over to the calculi presented here. We hope this ongoing work will lead to even greater insight of the theory and techniques of different tableau approaches for description logics and their implementation.

References

1. Alenda, R., Olivetti, N., Schwind, C., Tishkovsky, D.: Tableau calculi for CSL over min-spaces. In: Proc. CSL'10. LNCS, vol. 6247, pp. 52–66. Springer (2010)
2. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. *Studia Logica* 69(1), 5–40 (2001)
3. Bolander, T., Blackburn, P.: Termination for hybrid tableaux. *J. Logic Comput.* 17(3), 517–554 (2007)
4. Cialdea Mayer, M., Cerrito, S.: Nominal substitution at work with the global and converse modalities. In: Proc. AiML-8. pp. 57–74. College Publ. (2010)
5. Fitting, M.: Proof methods for modal and intuitionistic logics. Kluwer (1983)
6. Hintikka, J.: Model minimization: An alternative to circumscription. *J. Automat. Reason.* 4(1), 1–13 (1988)
7. Horrocks, I.: Using an expressive description logic: FaCT or fiction? In: Proc. KR-98. pp. 636–647. Morgan Kaufmann (1998)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Proc. KR 2006. pp. 57–67. AAAI Press (2006)
9. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. *J. Logic Comput.* 9(3), 385–410 (1999)
10. Horrocks, I., Sattler, U.: A tableau decision procedure for SHOIQ. *J. Automat. Reason.* 39(3), 249–276 (2007)
11. Schmidt, R.A., Tishkovsky, D.: A general tableau method for deciding description logics, modal logics and related first-order fragments. In: Proc. IJCAR'08. LNCS, vol. 5195, pp. 194–209. Springer (2008)
12. Schmidt, R.A., Tishkovsky, D.: Automated synthesis of tableau calculi. *Logical Methods in Comput. Sci.* 7(2), 1–32 (2011)
13. Schmidt, R.A., Tishkovsky, D.: Using tableau to decide description logics with full role negation and identity (2011), manuscript, <http://www.mettel-prover.org/papers/ALB0id.pdf>
14. Schmidt, R.A., Tishkovsky, D.: Using tableau to decide expressive description logics with role negation. In: Proc. ISWC+ASWC'07. pp. 438–451. Springer (2007)
15. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. Artificial Intelligence Res.* 12, 199–217 (2000)

Long Rewritings, Short Rewritings

S. Kikot¹, R. Kontchakov¹, V. Podolskii², and M. Zakharyashev¹

¹ Department of Computer Science and Information Systems
Birkbeck, University of London, U.K.

{kikot, roman, michael}@dcs.bbk.ac.uk

² Steklov Mathematical Institute, Moscow, Russia
podolskii@mi.ras.ru

1 Introduction

An ontology language \mathcal{L} is said to enjoy *FO-rewritability* if any conjunctive query (CQ) q over any ontology \mathcal{T} , given in \mathcal{L} , can be transformed into an FO-formula q' such that, for any data \mathcal{A} , all answers to q over the knowledge base $(\mathcal{T}, \mathcal{A})$ can be found by querying q' over \mathcal{A} using a standard relational database management system (RDBMS). Ontology languages with this property include the *OWL 2 QL* profile of *OWL 2*, which is based on description logics of the *DL-Lite* family [7, 16, 2], and fragments of Datalog[±] such as linear or sticky TGDs [5, 6]. Various rewriting techniques have been implemented in the systems QuOnto [1], REQUIEM [15], Presto [22], Nyaya [8], IQAROS³ and Quest⁴.

The idea of using languages with FO-rewritability for ontology-based data access (OBDA) relies on the empirical fact that RDBMSs are usually very efficient in practice. However, the first rewritings of CQs over *OWL 2 QL* ontologies [7, 15] turned out to be too lengthy even for modern RDBMSs. The attempts to employ various optimisation techniques still produced rewritings of exponential size in the worst case: $O((|\mathcal{T}| \cdot |q|)^{|q|})$ [22, 8, 20, 21]. The alternative two-step combined approach [14, 13]—first expand the data by applying the ontology axioms to the data and introducing (some of) the missing individuals, and only then rewrite the query over the expanded data—resulted in a simple polynomial rewriting only for the fragment of *OWL 2 QL* without role inclusions; for the full language, the rewriting remained exponential. Two seemingly contradictory results, presented at DL 2011, added more spice to the quest for short rewritings: [9] showed that one can construct, in polynomial time, a nonrecursive Datalog (NDL) rewriting for some fragments of Datalog[±] containing *OWL 2 QL*, while [11] argued that no FO-rewriting for *OWL 2 QL* can be constructed in polynomial time.

The aim of this paper is twofold. First, we investigate the worst-case size of FO- and NDL-rewritings for CQs over *OWL 2 QL* ontologies. We distinguish between ‘pure’ rewritings, which can use the signature of the original query and ontology as well as $=$, \neq (cf. [7]), and ‘impure’ rewritings, where other means such as new constants are allowed. Here is a summary of the obtained results:

³ <http://code.google.com/p/iqaros/>

⁴ <http://obda.inf.unibz.it/protege-plugin/quest/quest.html>

- (1) An exponential blow-up is unavoidable for pure positive existential (PE) rewritings and pure NDL-rewritings; pure FO-rewritings can blow-up super-polynomially unless $\text{NP} \subseteq \text{P/poly}$.
- (2) Pure NDL-rewritings are in general exponentially more succinct than pure PE-rewritings.
- (3) Pure FO-rewritings can be superpolynomially more succinct than pure PE-rewritings.
- (4) Impure PE- and NDL-rewritings can always be made polynomial, and so they are exponentially more succinct than pure PE- and NDL-rewritings, respectively.

(1)–(3) are proved by establishing connections between pure rewritings for CQs over *OWL 2 QL* ontologies and circuits for monotone Boolean functions. In a nutshell, we show that CQs and *OWL 2 QL* ontologies can encode such problems as the existence of a k -clique in a graph with n vertices whose edges are given by a single-element ABox. The polynomial PE-rewriting in (4) is similar to the NDL-rewriting of [9]: using two extra constants, $=$ and polynomially many new existentially quantified variables, one can guess a relevant part of the canonical model of \mathcal{T} in the rewritten query. The difference between the resulting impure PE-rewritings and exponential-size pure PE-rewritings is of the same kind as the difference between deterministic and nondeterministic Boolean circuits.

Our second aim is to analyse the causes behind long rewritings and whether they occur in real-world queries and ontologies. As a result, we suggest some short rewritings that cover most practical cases.

Omitted proofs can be found in [10] and the full version of [12].

2 Queries over *OWL 2 QL* Ontologies

The language of *OWL 2 QL* is defined by the following grammar:⁵

$$\begin{aligned}
 R & ::= P_i \mid P_i^-, \\
 B & ::= \perp \mid A_i \mid \exists R, \\
 C & ::= B \mid \exists R.B,
 \end{aligned}$$

where the A_i are concept names and the P_i are role names. An *OWL 2 QL TBox*, \mathcal{T} , is a finite set of *inclusions* of the form $B \sqsubseteq C$, $R_1 \sqsubseteq R_2$, $B_1 \sqcap B_2 \sqsubseteq \perp$ and $R_1 \sqcap R_2 \sqsubseteq \perp$. Note that concepts of the form $\exists R.B$ can only occur in the right-hand side of concept inclusions. An *ABox*, \mathcal{A} , is a finite set of *assertions* of the form $A_k(a_i)$ and $P_k(a_i, a_j)$, where a_i, a_j are individual names. \mathcal{T} and \mathcal{A} together form the *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The semantics is defined in the usual way, based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with domain $\Delta^{\mathcal{I}}$ and interpretation function $\cdot^{\mathcal{I}}$. The set of individuals in \mathcal{A} is denoted by $\text{ind}(\mathcal{A})$; $\mathcal{I}_{\mathcal{A}}$ is the interpretation with domain $\text{ind}(\mathcal{A})$ such that, for any concept or role E and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $\mathcal{I}_{\mathcal{A}} \models E(\mathbf{a})$ iff $E(\mathbf{a}) \in \mathcal{A}$. We write $E_1 \sqsubseteq_{\mathcal{T}} E_2$ if $\mathcal{T} \models E_1 \sqsubseteq E_2$; and we set $[E] = \{E' \mid E \sqsubseteq_{\mathcal{T}} E' \text{ and } E' \sqsubseteq_{\mathcal{T}} E\}$.

⁵ We do not consider data properties, attributes and role (ir)reflexivity constraints.

A *conjunctive query* (CQ) $\mathbf{q}(\mathbf{x})$ is a formula $\exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$, where φ is a conjunction of atoms of the form $A_k(t_1)$ and $P_k(t_1, t_2)$, and each t_i is a *term* (an individual or a variable from \mathbf{x}, \mathbf{y}). A tuple $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$ is a *certain answer* to $\mathbf{q}(\mathbf{x})$ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathbf{q}(\mathbf{a})$ for all $\mathcal{I} \models \mathcal{K}$; then we write $\mathcal{K} \models \mathbf{q}(\mathbf{a})$.

Query answering over OWL2QL KBs is based on the fact that, for any consistent KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, there is an interpretation $\mathcal{C}_{\mathcal{K}}$ such that, for all CQs $\mathbf{q}(\mathbf{x})$ and $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $\mathcal{K} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$. The interpretation $\mathcal{C}_{\mathcal{K}}$, called the *canonical model* of \mathcal{K} , can be constructed as follows. For each pair $[R], [B]$ with $\exists R.B$ in \mathcal{T} (we assume $\exists R$ is just a shorthand for $\exists R.\top$), we introduce a fresh symbol $w_{[RB]}$ and call it the *witness for $\exists R.B$* . We write $\mathcal{K} \models C(w_{[RB]})$ if $\exists R^- \sqsubseteq_{\mathcal{T}} C$ or $B \sqsubseteq_{\mathcal{T}} C$. Define a *generating relation*, \rightsquigarrow , on the set of these witnesses together with $\text{ind}(\mathcal{A})$ by taking:

- $a \rightsquigarrow w_{[RB]}$ if $a \in \text{ind}(\mathcal{A})$, $[R]$ and $[B]$ are $\sqsubseteq_{\mathcal{T}}$ -minimal such that $\mathcal{K} \models \exists R.B(a)$ and there is no $b \in \text{ind}(\mathcal{A})$ with $\mathcal{K} \models R(a, b) \wedge B(b)$;
- $w_{[R'B']}\rightsquigarrow w_{[RB]}$ if $u \rightsquigarrow w_{[R'B']}$, for some u , $[R]$ and $[B]$ are $\sqsubseteq_{\mathcal{T}}$ -minimal with $\mathcal{K} \models \exists R.B(w_{[R'B']})$ and it is not the case that $R' \sqsubseteq_{\mathcal{T}} R^-$ and $\mathcal{K} \models B(u)$.

If $a \rightsquigarrow w_{[R_1 B_1]} \rightsquigarrow \dots \rightsquigarrow w_{[R_n B_n]}$, $n \geq 0$, then we say that a *generates the path* $aw_{[R_1 B_1]} \dots w_{[R_n B_n]}$. Denote by $\text{path}_{\mathcal{K}}(a)$ the set of paths generated by a , and by $\text{tail}(\pi)$ the last element in $\pi \in \text{path}_{\mathcal{K}}(a)$. $\mathcal{C}_{\mathcal{K}}$ is defined by taking:

$$\begin{aligned} \Delta^{\mathcal{C}_{\mathcal{K}}} &= \bigcup_{a \in \text{ind}(\mathcal{A})} \text{path}_{\mathcal{K}}(a), & a^{\mathcal{C}_{\mathcal{K}}} &= a, \text{ for } a \in \text{ind}(\mathcal{A}), \\ A^{\mathcal{C}_{\mathcal{K}}} &= \{\pi \in \Delta^{\mathcal{C}_{\mathcal{K}}} \mid \mathcal{K} \models A(\text{tail}(\pi))\}, \\ P^{\mathcal{C}_{\mathcal{K}}} &= \{(a, b) \in \text{ind}(\mathcal{A}) \times \text{ind}(\mathcal{A}) \mid \mathcal{K} \models P(a, b)\} \cup \\ &\quad \{(\pi, \pi \cdot w_{[RB]}) \mid \text{tail}(\pi) \rightsquigarrow w_{[RB]}, R \sqsubseteq_{\mathcal{T}} P\} \cup \\ &\quad \{(\pi \cdot w_{[RB]}, \pi) \mid \text{tail}(\pi) \rightsquigarrow w_{[RB]}, R \sqsubseteq_{\mathcal{T}} P^-\}. \end{aligned}$$

Theorem 1 ([7, 13]). *For every OWL2QL KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, every CQ $\mathbf{q}(\mathbf{x})$ and every $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, $\mathcal{K} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$.*

Given a CQ $\mathbf{q}(\mathbf{x})$ and a TBox \mathcal{T} , a first-order formula $\mathbf{q}'(\mathbf{x})$, possibly with = and \neq , is called an *FO-rewriting for $\mathbf{q}(\mathbf{x})$ and \mathcal{T}* if, for any ABox \mathcal{A} and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{I}_{\mathcal{A}} \models \mathbf{q}'(\mathbf{a})$. If \mathbf{q}' is an FO-rewriting of the form $\exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$, where φ is built from atoms using only \wedge and \vee , then we call $\mathbf{q}'(\mathbf{x})$ a *positive existential rewriting for $\mathbf{q}(\mathbf{x})$ and \mathcal{T}* (or a *PE-rewriting*, for short). We say that \mathbf{q}' is *pure* if it does not contain constants that do not occur in \mathbf{q} (such constants are interpreted by fresh individuals added to $\mathcal{I}_{\mathcal{A}}$). The *size* $|\mathbf{q}'|$ of \mathbf{q}' is the number of symbols in \mathbf{q}' .

We also consider rewritings in the form of nonrecursive Datalog queries. We remind the reader that a *Datalog program*, Π , is a finite set of Horn clauses $\forall \mathbf{x} (A_1 \wedge \dots \wedge A_m \rightarrow A_0)$, where each A_i is an atom of the form $P(t_1, \dots, t_l)$ and each t_j is either a variable from \mathbf{x} or a constant. A_0 is called the *head* of the clause, and A_1, \dots, A_m its *body*. All variables occurring in the head must also occur in the body. A predicate P *depends* on a predicate Q in Π if Π contains a clause whose head is P and whose body contains Q . Π is called *nonrecursive* if

this dependence relation for Π is acyclic. A *nonrecursive Datalog query* consists of a nonrecursive Datalog program Π and a *goal* G , which is just a predicate. A tuple $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$ is a *certain answer* to (Π, G) over \mathcal{A} if $\Pi, \mathcal{A} \models G(\mathbf{a})$. The *size* $|\Pi|$ of Π is the number of symbols in it. We distinguish between *pure* and *impure* Datalog queries [3]. In a *pure query* (Π, G) , the clauses in Π do not contain constant symbols in their heads. One reason for considering only pure queries in OBDA is that impure ones can add new facts to the database that do not follow from the background ontology. Impure queries are known to be more succinct than pure ones.

Given a CQ $\mathbf{q}(\mathbf{x})$ and an OWL 2 QL TBox \mathcal{T} , a pure nonrecursive Datalog query (Π, G) is called a *nonrecursive Datalog rewriting for $\mathbf{q}(\mathbf{x})$ and \mathcal{T}* (or an *NDL-rewriting*, for short) if, for any ABox \mathcal{A} and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$ iff $\Pi, \mathcal{A} \models G(\mathbf{a})$. If Π does not contain constants that do not occur in \mathbf{q} then we say that the NDL-rewriting (Π, G) is *pure*.

3 Query Rewritings and Boolean Circuits

To establish results (1)–(3) on the size of rewritings mentioned in the introduction, we show how the problem of constructing circuits that compute monotone Boolean functions can be reduced to the problem of finding pure FO- and NDL-rewritings for CQs over OWL 2 QL ontologies.

Our reduction proceeds in three steps. First, we take any family f^1, f^2, \dots of *monotone* Boolean functions in NP, where $f^n: \{0, 1\}^n \rightarrow \{0, 1\}$, a polynomial p and a family $\mathbf{C}^1, \mathbf{C}^2, \dots$ of polynomial-size circuits such that $f^n(\boldsymbol{\alpha}) = 1$ iff $\mathbf{C}^n(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1$, for some $\boldsymbol{\beta} \in \{0, 1\}^{p(n)}$. Using the Tseitin transformation [23], we construct a polynomial-size CNF θ_{f^n} that computes f^n in the following sense:

Lemma 1. *If f^n is monotone then $\varphi_{f^n}^\alpha = (\bigwedge_{\alpha_j=0} \neg x_j) \wedge \theta_{f^n}$ is satisfiable iff $f^n(\boldsymbol{\alpha}) = 1$, for all $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \{0, 1\}^n$.*

Let φ_{f^n} be $\varphi_{f^n}^\alpha$ for $\boldsymbol{\alpha} = (0, \dots, 0)$. It should be clear that $\varphi_{f^n}^\alpha$ is obtained from φ_{f^n} by removing the clauses $\neg x_j$ for which the j th component of $\boldsymbol{\alpha}$ is 1.

The second step is to encode the φ_{f^n} by means of TBoxes \mathcal{T}_{f^n} and CQs \mathbf{q}_{f^n} . Let p_1, \dots, p_N be the propositional variables and C_1, \dots, C_d the clauses in φ_{f^n} . Then \mathcal{T}_{f^n} contains the following concept inclusions, for $1 \leq i \leq N$, $1 \leq j \leq d$:

$$\begin{aligned} A_{i-1} \sqsubseteq \exists P^- . X_i^\ell, \quad X_i^\ell \sqsubseteq A_i, \quad \text{for } \ell = 0, 1, \quad X_i^0 \sqsubseteq Z_{i,j} \quad \text{if } \neg p_i \in C_j, \\ Z_{i,j} \sqsubseteq \exists P . Z_{i-1,j}, \quad X_i^1 \sqsubseteq Z_{i,j} \quad \text{if } p_i \in C_j, \\ A_0 \sqcap A_i \sqsubseteq \perp, \quad A_0 \sqcap \exists P \sqsubseteq \perp, \quad A_0 \sqcap Z_{i,j} \sqsubseteq \perp, \quad \text{if } (i, j) \notin \{(0, 1), \dots, (0, n)\}, \end{aligned}$$

and the CQ is defined as follows:

$$\mathbf{q}_{f^n} = \exists \mathbf{y} \exists \mathbf{z} \left[A_0(y_0) \wedge \bigwedge_{i=1}^N P(y_i, y_{i-1}) \wedge \bigwedge_{j=1}^d \left(P(y_N, z_{N-1,j}) \wedge \bigwedge_{i=1}^{N-1} P(z_{i,j}, z_{i-1,j}) \wedge Z_{0,j}(z_{0,j}) \right) \right],$$

function $\text{CLIQUE}_{n,k}$ of $n(n-1)/2$ variables e_{ij} , $1 \leq i < j \leq n$, which returns 1 iff the graph with vertices $\{1, \dots, n\}$ and edges $\{\{i, j\} \mid e_{ij} = 1\}$ contains a k -clique. A series of papers, started by Razborov's [19], gave an exponential lower bound for the size of monotone circuits computing $\text{CLIQUE}_{n,k}$: $2^{\Omega(\sqrt{k})}$ for $k \leq \frac{1}{4}(n/\log n)^{2/3}$. For monotone formulas, an even better lower bound is known: $2^{\Omega(k)}$ for $k = 2n/3$ [18]. The question whether $\text{CLIQUE}_{n,k}$ can be computed by a polynomial-size circuit is equivalent to whether $\text{NP} \subseteq \text{P/poly}$.

To show (2), we use the function GEN_{n^3} of n^3 variables x_{ijk} , $1 \leq i, j, k \leq n$, defined as follows. We say that 1 *generates* $k \leq n$ if either $k = 1$ or, for some i and j such that $x_{ijk} = 1$, 1 generates both i and j . $\text{GEN}_{n^3}(x_{111}, \dots, x_{nnn})$ returns 1 iff 1 generates n . It is clearly a monotone function computable by polynomial-size monotone circuits. On the other hand, any monotone formula computing GEN_{n^3} is of size at least 2^{n^ε} , for some $\varepsilon > 0$ [17].

For (3), we use the function MATCHING_{2n} of n^2 variables e_{ij} , $1 \leq i, j \leq n$, which returns 1 iff there is a *perfect matching* in a bipartite graph G with vertices $\{v_1^1, \dots, v_n^1, v_1^2, \dots, v_n^2\}$ and edges $\{\{v_i^1, v_j^2\} \mid e_{ij} = 1\}$, i.e., a subset E of edges in G such that every node in G occurs exactly once in E . An exponential lower bound $2^{\Omega(n)}$ for the size of monotone formulas computing this function was obtained in [18]. However, MATCHING_{2n} is computable by non-monotone formulas of size $n^{O(\log n)}$ [4].

The exponential bounds for the size of rewritings can be reduced to *polynomial* by using two extra constants, say 0 and 1, which are included in the domain of every $\mathcal{I}_{\mathcal{A}}$ (see [9] and [10] for polynomial-size NDL- and PE-rewritings, respectively). As these constants may not occur in the original query and intended ABoxes, we call such rewritings *impure* (in [9] and [10], 0, 1, = and polynomially-many fresh existentially quantified variables are used to guess some part of the canonical model). The difference between pure and impure rewritings is similar to the difference between deterministic circuits and nondeterministic circuits with additional existentially quantified input variables. For example, $\text{CLIQUE}_{n,k}$ is computed by a polynomial-size monotone nondeterministic circuit, but not by a monotone deterministic circuit of polynomial size [19].

4 Why are Pure Rewritings so Long?

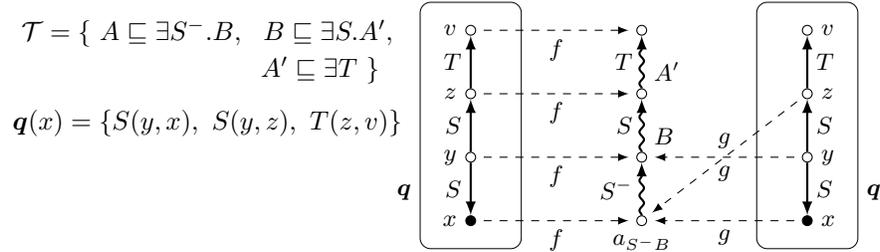
Let $\mathbf{q}(\mathbf{x}) = \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$ be a connected CQ and \mathcal{T} a TBox. Let *term* \mathbf{q} be the set of terms in \mathbf{q} . Denote by $\mathcal{C}_{\mathcal{T}}$ the disjoint union of the canonical models for consistent $(\mathcal{T}, \{R(a_{RB}, b_{RB}), B(b_{RB})\})$, in which the generating relation is extended with $a_{RB} \rightsquigarrow b_{RB}$. The *RB-subtree* of $\mathcal{C}_{\mathcal{T}}$ has root a_{RB} and consists of the full subtree of $\mathcal{C}_{\mathcal{T}}$ with root b_{RB} extended with the edge (a_{RB}, b_{RB}) . We also need the formulas

$$\text{ext}_C(x) = \bigvee_{A \sqsubseteq_{\mathcal{T}} C} A(x) \vee \bigvee_{\exists R \sqsubseteq_{\mathcal{T}} C} \exists y R(x, y), \quad \text{ext}_P(x, y) = \bigvee_{R \sqsubseteq_{\mathcal{T}} P} R(x, y),$$

for concepts C and role names P . Suppose $\mathcal{C}_{\mathcal{K}} \models^{\mathbf{a}} \varphi$, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where \mathbf{a} is an assignment of elements of $\Delta^{\mathcal{C}_{\mathcal{K}}}$ to the variables in φ under which $\mathbf{a}(x) \in \text{ind}(\mathcal{A})$

for all $x \in \mathbf{x}$. Consider an atom $P(t, t') \in \mathbf{q}$ with bound variables t, t' and assume that $\mathbf{a}(t_0) \in \text{ind}(\mathcal{A})$, for some $t_0 \in \text{term } \mathbf{q}$. The assignment \mathbf{a} can send t and t' to four different locations in $\mathcal{C}_{\mathcal{K}}$: (A) $\mathbf{a}(t), \mathbf{a}(t') \in \text{ind}(\mathcal{A})$; (B) $\mathbf{a}(t) \in \text{ind}(\mathcal{A})$, $\mathbf{a}(t') \notin \text{ind}(\mathcal{A})$; (B⁻) $\mathbf{a}(t) \notin \text{ind}(\mathcal{A})$, $\mathbf{a}(t') \in \text{ind}(\mathcal{A})$; (O) $\mathbf{a}(t), \mathbf{a}(t') \notin \text{ind}(\mathcal{A})$. Let us see how these alternatives can be reflected in a rewriting. In case (A) we have $\mathcal{C}_{\mathcal{K}} \models^{\mathbf{a}} P(t, t')$ iff $\mathcal{A} \models^{\mathbf{a}} \text{ext}_P(t, t')$. Case (B) is possible only if, for some concept $\exists R.B$, we have $\mathbf{a}(t) \rightsquigarrow w_{[RB]}$, $R \sqsubseteq_{\mathcal{T}} P$ and the atoms of \mathbf{q} ‘linked to’ t' can be mapped into the RB -subtree of $\mathcal{C}_{\mathcal{T}}$. To illustrate, consider an example.

Example 1. Let \mathcal{T} and \mathbf{q} be as in the picture below. The answer variable x must be mapped by \mathbf{a} to an ABox element. However, y can be mapped either to an ABox element or to the point $\mathbf{a}(x) \cdot w_{[S^-B]}$ provided that $\mathbf{a}(x)$ is an instance of $\exists S^-B$. In the latter case, we have two ways of mapping z : either to $\mathbf{a}(x) \cdot w_{[S^-B]}w_{[SA']}$, in which case we must set $\mathbf{a}(v) = \mathbf{a}(x) \cdot w_{[S^-B]}w_{[SA']}w_{[T\top]}$, or to $\mathbf{a}(x)$, provided that $\mathbf{a}(x)$ is an instance of $\exists T$. Thus we have two (partial) maps f and g from \mathbf{q} into the S^-B -subtree of $\mathcal{C}_{\mathcal{T}}$ shown below.



These observations motivate our key definition. Given a pair (t, t') of adjacent terms in \mathbf{q} , a *tree witness*⁶ for (t, t') is a homomorphism f from the query $\mathbf{q}_f = \{ E(s) \in \mathbf{q} \mid s \subseteq \text{dom } f, s \not\subseteq [t]_f \}$ to the RB -subtree of $\mathcal{C}_{\mathcal{T}}$, for some $\exists R.B$, such that $f(t) = a_{RB}$, $\text{dom } f$ is the smallest set containing t, t' for which $s' \in \text{dom } f$ whenever $S(s, s') \in \mathbf{q}$ with $s \in \text{dom } f \setminus [t]_f$, and all $s \in \text{dom } f \setminus [t]_f$ are bound variables in \mathbf{q} . Here \sim_f denotes the equivalence relation on $\text{dom } f$ defined by taking $s \sim_f s'$ iff $f(s) = f(s')$ and $[t]_f$ the equivalence class of t . In Example 1, f and g are two tree witnesses for (x, y) . Returning back to case (B), we can say now that there must exist a tree witness f for (t, t') such that $\mathbf{a}(t)$ satisfies the following *tree-witness formula* tw_f for f with $f(t) = a_{RB}$:

$$\text{tw}_f = \text{ext}_{\exists R.B}(t) \wedge \bigwedge_{s \in [t]_f} (t = s) \wedge \bigwedge_{E(s) \in \mathbf{q}, s \subseteq [t]_f} \text{ext}_{E(s)}.$$

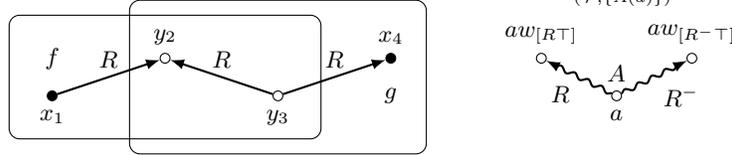
Case (B⁻) is symmetric, and in case (O) there must exist $S(s, s') \in \mathbf{q}$ for which (B) holds, with $P(t, t')$ being ‘covered’ by the tree witness for (s, s') .

This analysis suggests the following idea for a rewriting. We guess pairs of adjacent terms (t, t') in \mathbf{q} that will be mapped to edges of the tree part of $\mathcal{C}_{\mathcal{K}}$

⁶ A different notion of tree witness was used for $DL\text{-Lite}_{\text{horn}}^{\mathcal{N}}$ [13], where the structure of the canonical models ensured uniqueness of every tree witness.

and the tree witnesses that will cover them, as in cases (B), (B⁻) and (O). The part of \mathbf{q} that is not covered by these tree witnesses will be mapped to $\text{ind}(\mathcal{A})$, as in case (A). The query representing the guesses is then evaluated over $\mathcal{I}_{\mathcal{A}}$. The following example shows, however, that this idea needs a refinement.

Example 2. Let $\mathbf{q}(x_1, x_4) = \{R(x_1, y_2), R(y_3, y_2), R(y_3, x_4)\}$, shown below on the left, and $\mathcal{T} = \{A \sqsubseteq \exists R, A \sqsubseteq \exists R^-\}$.



Clearly, there is a tree witness f for (x_1, y_2) with $\text{dom } f = \{x_1, y_2, y_3\}$ and $[x_1]_f = \{x_1, y_3\}$, and a tree witness g for (x_4, y_3) with $\text{dom } g = \{x_4, y_3, y_2\}$ and $[x_4]_g = \{x_4, y_2\}$. Although these tree witnesses cover the whole query \mathbf{q} , they are only ‘realised,’ say, in the canonical model $\mathcal{C}_{(\mathcal{T}, \{A(a)\})}$ (shown above on the right) under *conflicting* maps: f sends x_1, y_3 to a and y_2 to $aw_{[R\top]}$, while g sends x_4, y_2 to a and y_3 to $aw_{[R^-\top]}$; in fact, $(\mathcal{T}, \{A(a)\}) \not\models \mathbf{q}(a, a)$.

Tree witnesses f and g , for (t, t') and (s, s') , respectively, are *compatible* if $\text{dom } f \cap \text{dom } g \subseteq [t]_f \cap [s]_g$. If f and g are incompatible and neither $\text{dom } f \subseteq \text{dom } g$ nor $\text{dom } g \subseteq \text{dom } f$, then we call f and g *conflicting* (e.g., f and g in Example 2). A set Ξ of tree witnesses is called *consistent* if all pairs of tree witnesses in Ξ are compatible. Let

$$\mathbf{q}_e(\mathbf{x}) = \text{detached}_{\mathbf{q}} \vee \bigvee_{\Xi \text{ consistent}} \exists \mathbf{y} \left(\bigwedge_{f \in \Xi} \text{tw}_f \wedge \bigwedge_{\substack{E(\mathbf{s}) \in \mathbf{q} \\ \mathbf{s} \not\subseteq \text{dom } f, \text{ for all } f \in \Xi}} \text{ext}_E(\mathbf{s}) \right),$$

where $\text{detached}_{\mathbf{q}} = \perp$ if $\mathbf{x} \neq \emptyset$; otherwise it is a disjunction of the sentences $\exists \mathbf{x} \text{ext}_{\exists R.B}(\mathbf{x})$ such that there is a homomorphism from \mathbf{q} to the RB -subtree of $\mathcal{C}_{\mathcal{T}}$. The next theorem shows that \mathbf{q}_e is a pure PE-rewriting for \mathbf{q} and \mathcal{T} :

Theorem 2. *For all \mathcal{A} and $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{I}_{\mathcal{A}} \models \mathbf{q}_e(\mathbf{a})$.*

Example 3. Let $\mathbf{q}(x) = \{R_i(x, y_i) \mid i \leq n\}$ and $\mathcal{T} = \{A_i \sqsubseteq \exists R_i \mid i \leq n\}$. Each pair (x, y_i) gives rise to one tree witness f_i with $\text{tw}_{f_i} = A_i(x) \vee \exists y R_i(x, y)$, and $\mathbf{q}_e = \bigvee_{N \subseteq [0, n]} \exists \mathbf{y} (\bigwedge_{i \in N} \text{tw}_{f_i} \wedge \bigwedge_{j \notin N} R_j(x, y_j))$.

The size of \mathbf{q}_e is $O((n_{\mathcal{T}, \mathbf{q}} + 1)^{|\mathbf{q}|} \cdot |\mathcal{T}| \cdot |\mathbf{q}|^2)$, where $n_{\mathcal{T}, \mathbf{q}}$ is the number of distinct tree witnesses. Now, we observe that if

(conf) *there are no conflicting tree witnesses for \mathbf{q} and \mathcal{T}*

then \mathbf{q}_e can be transformed to the query

$$\mathbf{q}_c(\mathbf{x}) = \text{detached}_{\mathbf{q}} \vee \exists \mathbf{y} \bigwedge_{\substack{\{t, t'\} \\ t, t' \text{ adjacent}}} \left[\bigwedge_{E(\mathbf{s}) \in \mathbf{q}, \mathbf{s} \subseteq \{t, t'\}} \text{ext}_E(\mathbf{s}) \vee \bigvee_{\substack{f \text{ is a tree witness} \\ t, t' \in \text{dom } f}} \text{tw}_f \right]$$

(if \mathbf{q} has no binary predicates then $\mathbf{q}_c = \mathbf{q}_e$).

Theorem 3. *If \mathcal{T} and \mathbf{q} satisfy **(conf)** then, for any ABox \mathcal{A} and $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{I}_{\mathcal{A}} \models \mathbf{q}_c(\mathbf{a})$.*

The size of \mathbf{q}_c is $O(n_{\mathcal{T}, \mathbf{q}} \cdot |\mathcal{T}| \cdot |\mathbf{q}|^2)$. So, if **(conf)** holds and $n_{\mathcal{T}, \mathbf{q}}$ is polynomial then \mathbf{q}_c is a *polynomial* pure PE-rewriting of \mathbf{q} and \mathcal{T} . For instance, the exponential \mathbf{q}_e of Example 3 reduces to polynomial $\mathbf{q}_c = \exists \mathbf{y} \bigwedge_{i \leq n} (R_i(x, y_i) \vee \text{tw}_{f_i})$. Note, however, that the CQs and TBoxes used in Section 3 generate *exponentially many* distinct tree witnesses.

We show now that, for a large class of CQs \mathbf{q} and TBoxes \mathcal{T} , all tree witnesses f for each (t, t') in \mathbf{q} (even if there are exponentially many of them) can be represented by a polynomial formula. Observe that each tw_f is determined by a concept $\exists R.B$ (such that the RB -subtree of $\mathcal{C}_{\mathcal{T}}$ contains the range of f) and the equivalence relation \sim_f on $\text{dom } f$. As the number of concepts $\exists R.B$ is linear in $|\mathcal{T}|$, the rewriting \mathbf{q}_c may be regarded polynomial if we show that all tree witnesses for each (t, t') have the same equivalence relation. For example, let $\mathbf{q}(x) = \{S(x, y), R(y, z_i) \mid 1 \leq i \leq n\}$ and $\mathcal{T} = \{A \sqsubseteq \exists S, \exists S^- \sqsubseteq \exists R.B_1, \exists S^- \sqsubseteq \exists R.B_2\}$. There are 2^n tree witnesses for (x, y) , as each z_i can be mapped either to a B_1 - or a B_2 -point in $\mathcal{C}_{\mathcal{T}}$, and yet they all define the same equivalence relation and the same tree-witness formula $\text{ext}_{\exists S}(x)$.

We formalise this intuition in the following definition. For a pair (t, t') of adjacent terms in \mathbf{q} , we call $\mathbf{f} = (\text{dom } \mathbf{f}, \sim_{\mathbf{f}})$ a *universal tree witness* for (t, t') if $\text{dom } \mathbf{f}$ is a subset of $\text{term } \mathbf{q}$ with $t, t' \in \text{dom } \mathbf{f}$ and $\sim_{\mathbf{f}}$ is an equivalence relation on $\text{dom } \mathbf{f}$ such that $\mathbf{q}_{\mathbf{f}} = \{E(\mathbf{s}) \in \mathbf{q} \mid \mathbf{s} \subseteq \text{dom } \mathbf{f}, \mathbf{s} \not\subseteq [t]_{\mathbf{f}}\} / \sim_{\mathbf{f}}$ is a tree-shaped query with root $[t]_{\mathbf{f}}$ and, for every tree witness g for (t, t') ,

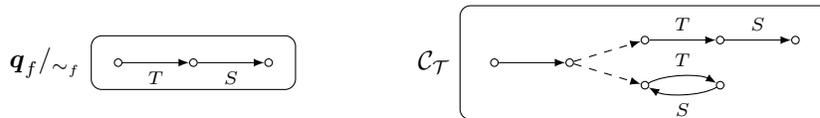
- $\text{dom } g = \text{dom } \mathbf{f}$ and there exists a homomorphism $h: \mathbf{q}_{\mathbf{f}} \rightarrow \mathcal{C}_{\mathcal{T}}$ that preserves the distance from the root and $g(\mathbf{s}) = h([s]_{\mathbf{f}})$, for every $\mathbf{s} \in \text{dom } \mathbf{f}$.

A universal tree witness \mathbf{f} for (t, t') is not a tree witness in the sense of our original definition, but rather a convenient structure representing all tree witnesses for (t, t') : we can merge the tree-witness formulas for (t, t') into one formula

$$\text{tw}_{\mathbf{f}} = \left[\bigvee_{\exists R.B \in \Phi_{\mathbf{f}}} \text{ext}_{\exists R.B}(t) \right] \wedge \bigwedge_{s \in [t]_{\mathbf{f}}} (t = s) \wedge \bigwedge_{E(\mathbf{s}) \in \mathbf{q}, \mathbf{s} \subseteq [t]_{\mathbf{f}}} \text{ext}_E(\mathbf{s}),$$

where $\Phi_{\mathbf{f}}$ is the set of concepts $\exists R.B$ such there is a homomorphism h from $\mathbf{q}_{\mathbf{f}}$ to the RB -subtree of $\mathcal{C}_{\mathcal{T}}$ with $h([t]_{\mathbf{f}}) = a_{RB}$.

We now identify a class of CQs and TBoxes for which a universal tree witness is unique (if exists) and can be constructed in time polynomial in $|\mathbf{q}|$ and $|\mathcal{T}|$. Intuitively, for each tree witness f , we disallow situations in which $\mathcal{C}_{\mathcal{T}}$ and the quotient $\mathbf{q}_{\mathbf{f}} / \sim_{\mathbf{f}}$ of $\mathbf{q}_{\mathbf{f}}$ simultaneously contain fragments of the form



We say that a role S is *forward* if $u \rightsquigarrow v$ for all $(u, v) \in S^{\mathcal{C}\tau}$. If neither S nor its inverse S^- is forward then S is said to be a *twisty role*. A tree witness f for (t, t') is called *perfect* if, for all $T(s_1, s_2), S(s_2, s_3) \in \mathbf{q}_f / \sim_f$ such that $s_2 \neq [t]_f$ and S is twisty, we have $\mathcal{C}\tau \not\models \text{inv}(T, S) \wedge \text{suc}(T, S)$, where

$$\begin{aligned} \text{suc}(T, S) &= \exists x, y, z (T(x, y) \wedge S(y, z) \wedge (x \neq z)), \\ \text{inv}(T, S) &= \exists x, y (T(x, y) \wedge S(y, x)). \end{aligned}$$

Lemma 5. *There is a polynomial-time algorithm which, given \mathbf{q} , \mathcal{T} and a pair (t, t') , checks whether all tree witnesses for (t, t') are perfect, and if this is the case, returns a unique universal tree witness $\mathbf{f}_{t, t'}$ for (t, t') .*

Note that even though there may be exponentially many tree witnesses for (t, t') , the algorithm checks whether they all are perfect in polynomial time. We are now in a position to define our *polynomial* PE-rewriting \mathbf{q}_p for \mathbf{q} and \mathcal{T} :

$$\mathbf{q}_p(\mathbf{x}) = \text{detached}_{\mathbf{q}} \vee \exists \mathbf{y} \bigwedge_{\substack{\{t, t'\} \\ t, t' \text{ adjacent}}} \left[\bigwedge_{E(s) \in \mathbf{q}, s \subseteq \{t, t'\}} \text{ext}_E(s) \vee \bigvee_{\substack{s, s' \text{ adjacent} \\ t, t' \in \text{dom } \mathbf{f}_{s, s'}}} \text{tw}_{\mathbf{f}_{s, s'}} \right].$$

Theorem 4. *If all tree witnesses for \mathbf{q} and \mathcal{T} are perfect and condition (conf) is satisfied then, for any ABox \mathcal{A} and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{I}_{\mathcal{A}} \models \mathbf{q}_p(\mathbf{a})$. Moreover, \mathbf{q}_p is constructed in time polynomial in $|\mathbf{q}|$ and $|\mathcal{T}|$.*

If \mathcal{T} does not contain any twisty roles, then all tree witnesses in any CQ \mathbf{q} over \mathcal{T} are perfect. On the other hand, all examples of conflicting tree witnesses above involve twisty roles. The following theorem shows that this is no accident:

Theorem 5. *Let \mathcal{T} be an OWL2QL ontology without twisty roles. Then, for any CQ \mathbf{q} , there are no conflicting tree witnesses for \mathbf{q} and \mathcal{T} . Thus, \mathbf{q}_p is a pure PE-rewriting for \mathbf{q} and \mathcal{T} and it can be constructed in polynomial time.*

Note that OWL2EL ontologies satisfy this condition, and so a polynomial rewriting similar to \mathbf{q}_p can also be used for CQ answering over such ontologies provided that the ABoxes are complete with respect to the ontologies [12].

5 Conclusions

As we saw in Section 3, pure PE- and NDL-rewritings of CQs over OWL2QL ontologies are of exponential size in the worst case. The analysis in Section 4 showed that the length of a rewriting is related to the number of tree witnesses in the query, which reflect how various parts of the query can be homomorphically mapped to the intensional tree part of the canonical model. Thus, a rewriting can be lengthy if the original query is sufficiently long and the intensional part of the canonical model for the ontology is sufficiently complex. We proved that by restricting the interaction between inverse roles and role inclusions in ontologies and queries, we can guarantee transparent polynomial rewritings. Moreover, as shown by a series of experiments [12], real-world ontologies and CQs contain very few tree witnesses, which are never in conflict, satisfy the above mentioned restrictions, and so enjoy pure, polynomial PE-rewritings.

References

1. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QUONTO: Querying ontologies. In: Proc. of the 20th Nat. Conf. on AI, AAAI. pp. 1670–1671 (2005)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)* 36, 1–69 (2009)
3. Benedikt, M., Gottlob, G.: The impact of virtual views on containment. *PVLDB* 3(1), 297–308 (2010)
4. Borodin, A., von zur Gathen, J., Hopcroft, J.E.: Fast parallel matrix and gcd computations. In: Proc. of FOCS. pp. 65–71 (1982)
5. Cali, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of PODS. pp. 77–86 (2009)
6. Cali, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. *PVLDB* 3(1), 554–565 (2010)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning* 39(3), 385–429 (2007)
8. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of the IEEE Int. Conf. on Data Engineering, ICDE (2011)
9. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive Datalog programs. In: Proc. of DL. vol. 745. CEUR-WS.org (2011)
10. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyashev, M.: Exponential lower bounds and separation for query rewriting. CoRR, arXiv:1202.4193, 2012.
11. Kikot, S., Kontchakov, R., Zakharyashev, M.: On (in)tractability of OBDA with OWL 2 QL. In: Proc. of DL. vol. 745. CEUR-WS.org (2011)
12. Kikot, S., Kontchakov, R., Zakharyashev, M.: Conjunctive query answering with OWL 2 QL. In: Proc. of KR. AAAI Press (2012) (see www.dcs.bbk.ac.uk/~kikot)
13. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to query answering in DL-Lite. In: Proc. of KR. AAAI Press (2010)
14. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic EL using a relational database system. In: Proceedings of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI 2009. pp. 2070–2075 (2009)
15. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-Lite. In: Proc. of DL. vol. 477. CEUR-WS.org (2009)
16. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. *J. on Data Semantics X*, 133–173 (2008)
17. Raz, R., McKenzie, P.: Separation of the monotone NC hierarchy. In: Proc. of FOCS. pp. 234–243 (1997)
18. Raz, R., Wigderson, A.: Monotone circuits for matching require linear depth. *J. ACM* 39(3), 736–744 (1992)
19. Razborov, A.: Lower bounds for the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR* 281(4), 798–801 (1985)
20. Rodríguez-Muro, M., Calvanese, D.: Dependencies to optimize ontology based data access. In: Proc. of DL. vol. 745. CEUR-WS.org (2011)
21. Rodríguez-Muro, M., Calvanese, D.: Semantic index: Scalable query answering without forward chaining or exponential rewritings. In: Proc. of ISWC (2011)
22. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proc. of the 12th Int. Conf. KR. AAAI Press (2010)
23. Tseitin, G.: On the complexity of derivation in propositional calculus. In: *Automation of Reasoning 2: Classical Papers on Computational Logic 1967–1970* (1983)

Cost Based Query Ordering over OWL Ontologies

Ilianna Kollia^{1,2} and Birte Glimm¹

¹ Ulm University, Germany, {birte.glimm, ilianna.kollia}@uni-ulm.de

² National Technical University of Athens, Greece

1 Introduction

Query answering—the computation of answers to users’ queries w.r.t. ontologies and data—is an important task provided by many description logic (DL) reasoners. Although much effort has been spent on optimizing the ‘reasoning’ part of query answering, i.e., the extraction of the individuals that satisfy a concept or role atom, less attention has been given to optimizing the actual query answering part when ontologies in expressive languages are used.

In the context of databases or triple stores, cost-based ordering techniques for finding an optimal or near optimal join ordering have been widely applied [9, 10]. These techniques involve the maintenance of a set of statistics about relations and indexes, e.g., number of pages in a relation, number of pages in an index, number of distinct values in a column, together with formulas for the estimation of the selectivity of predicates and the estimation of the CPU and I/O costs of query execution that depends amongst others, on the number of pages that have to be read from or written to secondary memory. The formulas for the estimation of selectivities of predicates (result output size of query atoms) estimate the data distributions using histograms, parametric or sampling methods or combinations of them.

In the context of ontologies, the formulas should be extended to take into account another important cost component, i.e., the cost of executing specific reasoner tasks such as entailment checks or instance retrievals. It is much more difficult to estimate this cost precisely before query evaluation as this cost varies and takes values from a wide range. For example, *SROIQ* has a worst case complexity of 2-NExpTime [4] and typical implementations are not worst case optimal. The hypertableau satisfiability checking algorithm for *SROIQ* that we use in this paper has a worst-case complexity of 3-NExpTime in the size of the ontology [7, 4].³ Instead, a subset of possible mappings for the variables of a query can be sampled and, based on the statistics extracted from these samples, the most efficient join ordering can be estimated. This, however, requires that the samples are selected according to an ontology based criterion and not randomly. Preliminary efforts for finding good join ordering in knowledge bases have already been made [8].

In this paper we address the issue of query atom ordering, which constitutes an optimization task, for conjunctions of instance queries issued over ontologies with expressivity up to *SROIQ*. The optimization goal is to find the execution plan (an order for query atoms) which leads to the most efficient execution of the query. This involves

³ The 2-NExpTime result for *SROIQ+* increases to 3-NExpTime when adding role chains [4]

the minimization of the number of needed reasoning tasks and the size of intermediate results. The execution plan which satisfies the above property is determined by means of a cost function that assigns costs to query atoms within an execution plan. This cost function is based on heuristics and summaries for statistics about the data which are extracted from a DL reasoner model. We explore static and dynamic algorithms that greedily explore the execution plan search space to determine an optimal or near optimal execution plan. Static ordering refers to the finding of a join order before query evaluation starts whereas dynamic ordering determines the ordering of query atoms during query evaluation taking advantage of already computed query atom results.

2 Preliminaries

In this section we briefly present an overview of the model building tableau and hypertableau calculi and give an introduction to conjunctive instance queries.

It is known that checking whether an individual s_0 (pair of individuals $\langle s_0, s_1 \rangle$) is an instance (are instances) of a concept C (role R) w.r.t. an ontology \mathcal{O} is equivalent to checking whether $\mathcal{O} \cup \{-C(c_0)\}$ ($\mathcal{O} \cup \{(\forall R. \neg\{s_1\})(s_0)\}$) is inconsistent w.r.t. \mathcal{O} . In order to perform this action, most DL reasoners use a model construction calculus such as tableau or hypertableau. In the remainder, we focus on the hypertableau calculus [7], but a tableau calculus could equally be used and we state how our results can be transferred to tableau calculi. The hypertableau calculus starts from an initial set of assertions and by applying derivation rules it tries to construct (an abstraction of) a model of \mathcal{O} . Derivation rules usually add new concept and role assertion axioms, they may introduce new individuals, they can be nondeterministic, leading to the need to choose between several alternative assertions to add or they can lead to a clash when a contradiction is detected. To show that an ontology \mathcal{O} is (in)consistent, the hypertableau calculus constructs a *derivation*, i.e., a sequence of sets of assertions A_0, \dots, A_n , such that A_0 contains all assertions in \mathcal{O} , A_{i+1} is the result of applying a derivation rule to A_i and A_n is the final set of assertions where no more rules are applicable. If a derivation exists such that A_n does not contain a clash, then \mathcal{O} is consistent and A_n is called a *pre-model* of \mathcal{O} . Otherwise \mathcal{O} is inconsistent. Each assertion in a set of assertions A_i is derived either *deterministically* or *nondeterministically*. An assertion is derived deterministically if it is derived by the application of a deterministic derivation rule from assertions that were all derived deterministically. Any other derived assertion is derived nondeterministically. It is easy to know whether an assertion was derived deterministically or not because of the dependency directed backtracking that most (hyper)tableau reasoners employ. In the pre-model, each individual s_0 is assigned a label $\mathcal{L}(s_0)$ representing the concepts it is (non)deterministically an instance of and each pair of individuals $\langle s_0, s_1 \rangle$ is assigned a label $\mathcal{L}(\langle s_0, s_1 \rangle)$ representing the roles through which individual s_0 is (non)deterministically related to individual s_1 .

Definition 1. Let $S = (N_C, N_R, N_I)$ be a signature of an ontology \mathcal{O} , N_V a countably infinite set of variables disjoint from N_C, N_R and N_I and $S_{\mathcal{O}} = (C_{\mathcal{O}}, R_{\mathcal{O}}, I_{\mathcal{O}})$ the restriction of S to terms that occur in \mathcal{O} . A term t is an element from $N_V \cup N_I$. Let $C \in C_{\mathcal{O}}$ be a concept, $r \in R_{\mathcal{O}}$ a role, and $t, t' \in I_{\mathcal{O}} \cup N_V$ terms. An atom is an expression

$C(t)$ or $r(t, t')$ and we refer to these types of atoms as concept and role atoms, respectively. A query q is a non-empty set of atoms. We use $\text{Vars}(q)$ to denote the set of variables occurring in q , $\text{Inds}(q)$ to denote the set of individual names occurring in q , and $\text{Terms}(q) = \text{Vars}(q) \cup \text{Inds}(q)$ for the set of terms in q . We use $|q|$ to denote the number of atoms in q .

Let $q = \{at_1, \dots, at_n\}$ be a query. A mapping μ for q over \mathcal{O} is a total function $\mu: \text{Terms}(q) \rightarrow I_{\mathcal{O}}$ such that $\mu(a) = a$ for each $a \in \text{Inds}(q)$. The set Γ_q of all possible mappings for q is defined as $\Gamma_q := \{\mu \mid \mu \text{ is a mapping for } q\}$. A solution mapping μ for q over \mathcal{O} is a mapping such that $\mathcal{O} \models C(\mu(t))$ for each concept atom $C(t) \in q$ and $\mathcal{O} \models r(\mu(t), \mu(t'))$ for each role atom $r(t, t') \in q$.

According to the above definition, we deal with conjunctive queries with only distinguished variables. Without loss of generality, we assume that queries are connected. In case they are not, the connected components of a query can be evaluated independently and the results of these evaluations can be combined in the end [1].

3 Extracting Individual Information from Reasoner Models

The first step in the ordering of query atoms is the extraction of statistics, exploiting information generated by reasoners.

As has been mentioned in Section 2, if an ontology is consistent and a pre-model is constructed, individuals are assigned labels in this pre-model. These labels can provide us with information about the concepts the individuals belong to or the roles in which they participate. We exploit this information similarly as was suggested for determining known (non-)subsumers for classes during classification [2]. In the hypertableau calculus, the following two properties hold for each ontology \mathcal{O} and each constructed pre-model A_n for \mathcal{O} :

- (P1) for each atomic concept C (role R), each individual s_0 (each pair of individuals $\langle s_1, s_2 \rangle$) in A_n , if $C \in \mathcal{L}_{A_n}(s_0)$ ($R \in \mathcal{L}_{A_n}(\langle s_1, s_2 \rangle)$) and the assertion $C(s_0)$ ($R(s_1, s_2)$) was derived deterministically, then it holds $\mathcal{O} \models C(s_0)$ ($\mathcal{O} \models R(s_1, s_2)$).
- (P2) for an arbitrary individual s_0 in A_n (an arbitrary pair of individuals $\langle s_1, s_2 \rangle$ in A_n) and an arbitrary atomic concept C (simple role R), if $C \notin \mathcal{L}_{A_n}(s_0)$ ($R \notin \mathcal{L}_{A_n}(\langle s_1, s_2 \rangle)$), then $\mathcal{O} \not\models C(s_0)$ ($\mathcal{O} \not\models R(s_1, s_2)$).

We use these properties to extract information from the pre-model of a satisfiable ontology \mathcal{O} as outlined in Algorithm 1. In our implementation we use a more complicated procedure to only store the direct types of each individual. The information we extract involves the maintenance of the sets of known and possible instances for all atomic concepts of \mathcal{O} . The known instances of a concept C ($K[C]$) are the individuals that can be safely considered instances of the concept according to the pre-model, i.e., the individuals of the assertions referring to concept C that have been derived deterministically. The possible instances of a concept C ($P[C]$) are the individuals of the assertions referring to C that have been derived nondeterministically. These individuals require costly consistency checks in order to decide whether they are real instances of the respective concept.

Algorithm 1 initializeKnownAndPossibleConceptInstances

Require: a consistent *SR*OIQ ontology \mathcal{O} to be queried

Ensure: sets $K[C]$ ($P[C]$) of known (possible) instances for each concept C of \mathcal{O} are computed

```
1:  $A_n := buildModelFor(\mathcal{O})$ 
2: for all  $ind \in I_{\mathcal{O}}$  do
3:   for all  $C \in \mathcal{L}_{A_n}(ind)$  do
4:     if  $C$  was derived deterministically then
5:        $K[C] := K[C] \cup \{ind\}$ 
6:     else
7:        $P[C] := P[C] \cup \{ind\}$ 
8:     end if
9:   end for
10: end for
```

The procedure to find the known and possible instances of simple and complex (transitive roles or roles having a transitive subrole) roles or, given an individual, the known and possible role successors or predecessors, can be defined similarly. In the case of complex roles, however, before the *buildModelFor* procedure is applied, \mathcal{O} is expanded with additional axioms that capture the semantics of the transitive relations since (hyper)tableau reasoners typically do not deal with transitivity directly [7]. In particular, for each individual ind and each complex role p , the new concepts C_{ind}^p and C_{ind} are created and the axioms $C_{ind}(ind)$ and $C_{ind} \sqsubseteq \forall p.C_{ind}^p$ are added to \mathcal{O} . Intuitively, the consequent application of the transitivity encoding [7] produces axioms $C_{ind}^p(s)$ that propagate to each individual s that is reachable from ind via a p -chain. The known and possible p -successors for ind can then be determined from concept assertions $C_{ind}^p(s)$ in the pre-model.

The technique presented in this paper can be used with any (hyper)tableau calculus for which properties (P1) and (P2) hold. All (hyper)tableau calculi used in practice that we are aware of satisfy property (P1). Pre-models produced by tableau algorithms as presented in the literature also satisfy property (P2); however, commonly used optimizations, such as lazy unfolding, can compromise property (P2), which we illustrate with the following example. Let us assume we have an ontology \mathcal{O} containing the axioms $A \sqsubseteq \exists R.(C \sqcap D)$, $B \equiv \exists R.C$ and $A(a)$. It is obvious that in this ontology A is a subconcept of B (hence $\mathcal{O} \models B(a)$) since every individual that is R -related to an individual that is an instance of the intersection of C and D is also R -related to an individual that is an instance of the concept C . However, even though the assertion $A(a)$ occurs in the ABox, the assertion $B(a)$ is not added in the pre-model when we use lazy unfolding. With lazy unfolding, instead of treating $B \equiv \exists R.C$ as two disjunctions $\neg B \sqcup \exists R.C$ and $B \sqcup \forall R.(\neg C)$ as is typically done for general concept inclusion axioms (GCIs), B is only lazily unfolded into its definition $\exists R.C$ once B occurs in the label of an individual. Thus, although $(\exists R.(C \sqcap D))(a)$ would be derived, this does not lead to the addition of $B(a)$.

Nevertheless, most (if not all) implemented calculi produce pre-models that satisfy at least the following weaker property:

(P3) for an arbitrary individual s_0 in A_n and an arbitrary concept C where C is primitive in \mathcal{O} ,⁴ if $C \notin \mathcal{L}_{A_n}(s_0)$, then $\mathcal{O} \not\models C(s_0)$.

Hence, properties (P2) and (P3) can be used to extract (non-)instance information from pre-models. For tableau calculi that only satisfy (P3), Algorithm 1 can be modified accordingly. In particular, for each non-primitive concept C in \mathcal{O} we need to add to $P[C]$ the individuals in \mathcal{O} that do not include the concept C in their label.

Even though the proposed technique for determining known and possible instances of concepts and roles can be used in the same way with both tableau and hypertableau reasoners, the effect that it will have when tableau algorithms are used is less intense. This happens because tableau algorithms often introduce more nondeterminism than hypertableau. In particular, in tableau algorithms a disjunction is added to each individual for each GCI in \mathcal{O} and, when optimizations such as lazy unfolding are used, these compromise property (P2) and we have to use the weaker property (P3) or even consider all concepts that do not occur in the label of an individual as possible types, which results in less accurate statistics.

4 Query Answering and Query Atom Ordering

In this section we describe two different algorithms (a static and a dynamic one) for ordering the atoms of a query based on some costs and then we deal with the formulation of these costs. We first introduce the abstract graph representation of a query q by means of a labeled graph G_q on which we define the computed statistical costs.

Definition 2. A query join graph G_q for a query q is a tuple (V, E, E_L) , where

- $V = q$ is a set of vertices (one for each query atom);
- $E \subseteq V \times V$ is a set of edges such that $\langle at_1, at_2 \rangle \in E$ iff $\text{Vars}(at_1) \cap \text{Vars}(at_2) \neq \emptyset$ and $at_1 \neq at_2$;
- E_L is a function that assigns a set of variables to each $\langle at_1, at_2 \rangle \in E$ such that $E_L(at_1, at_2) = \text{Vars}(at_1) \cap \text{Vars}(at_2)$.

In the remainder we use q for a query $\{at_1, \dots, at_n\}$, G_q for the according query join graph and Ω_q for the solution mappings of q . Our goal is to find a query execution plan, which determines the evaluation order for atoms in q . Since the number of possible execution plans is of order $|q|!$, the ordering task quickly becomes impractical. In the following, we focus on greedy algorithms for determining an execution order, which prune the search space considerably. Roughly speaking, we proceed as follows: We define a cost function, which consists of two components: an estimate for the reasoning costs and an estimate for the intermediate result size. Both components are combined to induce an order among query atoms. In this paper, we simply build the sum of the two cost components, but different combinations such as a weighted sum of the two values could also be used. For the query plan construction we distinguish *static* from *dynamic planning*. For the former, we start constructing the plan by adding a minimal

⁴ A concept C is considered primitive in \mathcal{O} if \mathcal{O} is unfoldable and it contains no axiom of the form $C \equiv E$

atom according to the order. Variables from this atom are then considered bound, which changes the cost function and might induce a different order among the remaining query atoms. Considering the updated order, we again select the minimal query atom that is not yet in the plan and update the costs. This process continues until the plan contains all atoms. Once a complete plan has been determined the atoms are evaluated. The dynamic case differs in that after selecting an atom for the plan, we immediately determine the solutions for the chosen atom, which are then used to update the cost function. Dynamic ordering is a costly but accurate procedure. Sampling techniques can be used so that not all mappings are used for the update of the cost function but only a subset of them. In Section 5 we show that random sampling is not adequate and a more sophisticated sampling criterion is needed. However, the definition of such criterion is out of the scope of the current paper. We now make the process of query plan construction more precise, but we leave the exact details of defining the cost function and the ordering it induces to later.

Definition 3. A static (dynamic) cost function w.r.t. q is a function $s: q \times 2^{\text{Vars}(q)} \rightarrow \mathbb{R} \times \mathbb{R}$ ($d: q \times 2^{I^q} \rightarrow \mathbb{R} \times \mathbb{R}$). The two costs are combined to yield a static ordering \leq_s (a dynamic ordering \leq_d), which is a total order over the atoms of q .

An execution plan for q is a duplicate-free sequence of query atoms from q . The initial execution plan is the empty sequence and a complete execution plan is a sequence containing all atoms of q . For $P_i = (at^1, \dots, at^i)$ with $i < n$ an execution plan for q with query join graph $G_q = (V, E, E_L)$, we define the potential next atoms q_i for P_i w.r.t. G_q as $q_i = q$ for P_i the initial execution plan and $q_i = \{at \mid \langle at', at \rangle \in E, at' \in \{at^1, \dots, at^i\}, at \in q \setminus \{at^1, \dots, at^i\}\}$ otherwise. The static (dynamic) ordering induces an execution plan $P_{i+1} = (at^1, \dots, at^i, at^{i+1})$ with $at^{i+1} \in q_i$ and $at^{i+1} \leq_s at$ ($at^{i+1} \leq_d at$) for each $at \in q_i$ such that $at \neq at^{i+1}$.

For $i > 0$, the set of potential next atoms only contains atoms that are connected to an atom that is already in the plan since unconnected atoms will cause an unnecessary blowup of the number of intermediate results. Let $P_i = (at_1, \dots, at_i)$ with $i \leq n$ be an execution plan for q . The procedure we follow to find the solution mappings Ω_i for P_i is recursively defined as follows: Initially, our solution set contains only the identity mapping $\Omega_0 = \{\mu_0\}$, which does not map any variable to any value. Assuming that we have evaluated the sequence $P_{i-1} = (at_1, \dots, at_{i-1})$ and we have found the set of solution mappings Ω_{i-1} , in order to find the solution mappings Ω_i of P_i , we use instance retrieval tasks of reasoners to extend the mappings in Ω_{i-1} to cover the new variables of at_i or the entailment check service of reasoners if at_i does not contain new variables. A detailed description of the method we are using for the evaluation of an execution plan together with optimizations can be found in our previous work [5].

We now define the cost functions s and d more precisely, which estimate the cost of the required reasoner operations and the estimated result output size of evaluating a query atom. The intuition behind the estimated value of the reasoner operation costs (the functions' first component) is that the evaluation of possible instances is much more costly than the evaluation of known instances since possible instances require expensive consistency checks whereas known instances require cheap cache lookups. The estimated result size (the functions' second component) takes into account the number

of known and possible instances and the probability that possible instances are actual instances. The static cost function has more cases since for atoms we might only know that their variables are bound, without knowing to which individuals they are bound to. The functions depend on several factors:

- $K[C]$ ($K[R]$) and $P[C]$ ($P[R]$) for known and possible instances of a concept C (a role R) from Section 3
- $\text{sucK}[R] := \{i \mid \exists j. \langle i, j \rangle \in K[R]\}$ ($\text{preK}[R] := \{i \mid \exists j. \langle j, i \rangle \in K[R]\}$) for the individuals with known successors (predecessors) for a role R . We define analogous functions $\text{sucP}[R]$ and $\text{preP}[R]$ for the individuals with possible successors and predecessors for a role R
- $\text{sucK}[R, a] := \{i \mid \langle a, i \rangle \in K[R]\}$ ($\text{preK}[R, a] := \{i \mid \langle i, a \rangle \in K[R]\}$) for known R-successors (R-predecessors) of an individual a . We define analogous functions $\text{sucP}[R, a]$ and $\text{preP}[R, a]$ for possible R-successors and R-predecessors of an individual a
- C_L for the cost of a cache lookup in the reasoner’s internal structures
- C_E for the cost of an entailment check
- P_{IS} for the possible instance success, i.e, an estimate for percentage of possible instances that are actual instances

The values C_L , C_E and P_{IS} are determined experimentally. The time needed for a cache lookup is much less than the time needed for an entailment check with the difference between the two depending on the ontology and even within an ontology on the queried concept (role). The two costs (C_L and C_E) were determined by taking the average time of the previous performed checks (lookup or entailment). In the case of C_E , we multiply this number with the depth of the concept (role) hierarchy. The depth of the concept (role) hierarchy should be taken into account for the estimation of C_E since we only store the direct types of each individual (roles in which each individual participates). In order to find the instances of a concept (role), we may need to check all its subconcepts (subroles) that contain possible instances. The possible instance success, P_{IS} , was determined by testing several ontologies and checking how many of the initial possible instances were real ones, which was around 50% in nearly all ontologies.

In the following, we use a, b for individual names and x, y for variables. We first define the static cost function s , which takes a pair $\langle at(\vec{t}), VarsB \rangle$ as input, where $at(\vec{t})$ is a query atom and $VarsB$ is the set of (bound) variables from $Vars(at(\vec{t}))$, and returns a pair of real numbers as follows:

- for $\langle at(\vec{t}), VarsB \rangle \in \{\langle C(x), \emptyset \rangle, \langle R(x, y), \emptyset \rangle\}$

$$\langle |K[at]| \cdot C_L + |P[at]| \cdot C_E, |K[at]| + P_{IS} \cdot |P[at]| \rangle \quad (1)$$

- for $\langle at(\vec{t}), VarsB \rangle \in \{\langle R(a, x), \emptyset \rangle\}$

$$\langle |\text{sucK}[at, a]| \cdot C_L + |\text{sucP}[at, a]| \cdot C_E, |\text{sucK}[at, a]| + P_{IS} \cdot |\text{sucP}[at, a]| \rangle \quad (2)$$

- for $\langle at(\vec{t}), VarsB \rangle \in \{\langle R(x, a), \emptyset \rangle\}$ we use preK (preP) instead of sucK (sucP) in (2)

Table 1. Query Ordering Example

	Atom Sequences	Known Instances	Possible Instances	Real from Possible Instances
1	C(x)	200	350	200
2	R(x,y)	200	200	50
3	D(y)	700	600	400
4	R(x,y), C(x)	100	150	100
5	R(x,y), D(y)	50	50	40
6	R(x,y), D(y), C(x)	45	35	25
7	R(x,y), C(x), D(y)	45	40	25

– for $\langle at(\vec{t}), VarsB \rangle \in \{\langle C(a), \emptyset \rangle, \langle R(a, b), \emptyset \rangle\}$

$$\begin{aligned}
 & \langle C_L, 1 \rangle \text{ if } \vec{t} \in K[at] \\
 & \langle C_E, P_{IS} \rangle \text{ if } \vec{t} \in P[at] \\
 & \langle C_L, 0 \rangle \text{ otherwise}
 \end{aligned} \tag{3}$$

– for $\langle at(\vec{t}), VarsB \rangle \in \{\langle C(x), \{x\} \rangle, \langle R(x, y), \{x, y\} \rangle, \langle R(a, x), \{x\} \rangle, \langle R(x, a), \{x\} \rangle\}$

$$\left\langle \frac{|K[at]|}{|I_O|} \cdot C_L + \frac{|P[at]|}{|I_O|} \cdot C_E, \frac{|K[at]| + P_{IS} \cdot |P[at]|}{|I_O| \cdot |I_O|} \right\rangle \tag{4}$$

– for $\langle at(\vec{t}), VarsB \rangle = \langle R(x, y), \{x\} \rangle$

$$\left\langle \frac{|K[at]|}{|\text{suc}K[at]|} \cdot C_L + \frac{|P[at]|}{|\text{suc}P[at]|} \cdot C_E, \frac{|K[at]|}{|\text{suc}K[at]|} + \frac{|P[at]|}{|\text{suc}P[at]|} \cdot P_{IS} \right\rangle \tag{5}$$

– for $\langle at(\vec{t}), VarsB \rangle = \langle R(x, y), \{y\} \rangle$ we use preK (preP) instead of sucK (sucP) in (5)

The dynamic cost function d is based on the static function s , but only applies to cases (1), (2) and (3). The function takes a pair $\langle at(\vec{t}), \Omega \rangle$ as input, where $at(\vec{t})$ is a query atom and Ω is the set of solution mappings for the atoms that have already been evaluated, and returns a pair of real numbers using matrix addition as follows:

$$d(at(\vec{t}), \Omega) = \sum_{\mu \in \Omega} s(\mu(at(\vec{t})), \emptyset)$$

A motivating example showing the difference between static and dynamic ordering and justifying why dynamic ordering can be beneficial in our setting is shown below. Let us assume that a query q consists of the three query atoms: $C(x)$, $R(x, y)$, $D(y)$. Table 1 gives information about the known and possible instances of these atoms within a sequence. In particular, the first column enumerates possible execution sequences $S_i = (at_1, \dots, at_i)$ for the atoms of q . Column 2 (3) gives the number of mappings to known (possible) instances of at_i (i.e., the number of known (possible) instances of at_i that satisfy at the same time the atoms (at_1, \dots, at_{i-1})). Column 4 gives the number of possible instances of at_i from Column 3 that are real instances (that belong to Ω_i). Let

us assume that we have 10,000 individuals in our ontology \mathcal{O} . We will now explain what the formulas described above are doing. We assume that $C_L \leq C_E$ which is always the case since a cache lookup is less expensive than a consistency check. In both techniques (static and dynamic) the atom $R(x, y)$ will be chosen first since it has the least number of possible instances (200) while it has the same (or smaller) number of known instances (200) with the other atoms ($s(R(x, y), \emptyset) = d(R(x, y), \{\mu_0\}) = \langle 200 \cdot C_L + 200 \cdot C_E, 200 + P_{IS} \cdot 200 \rangle$, $s(C(x), \emptyset) = d(C(x), \{\mu_0\}) = \langle 200 \cdot C_L + 350 \cdot C_E, 200 + P_{IS} \cdot 350 \rangle$, $s(D(y), \emptyset) = d(D(y), \{\mu_0\}) = \langle 700 \cdot C_L + 600 \cdot C_E, 700 + P_{IS} \cdot 600 \rangle$). Then, in the case of static ordering, after $R(x, y)$, the atom $C(x)$ is chosen since C has less possible (and known) instances than D (350 versus 600). Indeed, $s(C(x), \{x\}) = \langle \frac{200}{10,000} \cdot C_L + \frac{350}{10,000} \cdot C_E, \frac{200+350 \cdot P_{IS}}{10^8} \rangle$, $s(D(y), \{y\}) = \langle \frac{700}{10,000} \cdot C_L + \frac{600}{10,000} \cdot C_E, \frac{700+600 \cdot P_{IS}}{10^8} \rangle$. Hence, the order of evaluation in this case will be $P = (R(x, y), C(x), D(y))$ leading to 200(row 2) + 150(row 4) + 40(row 7) entailment checks. In the dynamic case, after the evaluation of $R(x, y)$, which gives a set of solutions \mathcal{Q}_1 , the atom $D(y)$ has fewer known and possible instances (50 known and 50 possible) than the atom $C(x)$ (100 known and 150 possible) and, hence, a lower cost. Indeed, $d(D(y), \mathcal{Q}_1) = \langle 50 \cdot C_L + 150 \cdot C_L + 50 \cdot C_E, 50 + 0 + 50 \cdot P_{IS} \rangle$, $d(C(x), \mathcal{Q}_1) = \langle 100 \cdot C_L + 0 \cdot C_L + 150 \cdot C_E, 100 + 0 + 150 \cdot P_{IS} \rangle$. Therefore, atom $D(y)$ will be chosen next leading to the execution of the query atoms in the order $P = (R(x, y), D(y), C(x))$ and the execution of 200(row 2) + 50(row 5) + 35(row 6) entailment checks.

5 Evaluation

We tested our ordering techniques with the Lehigh University Benchmark (LUBM) [3] as a case where no disjunctive information is present and with the more expressive University Ontology Benchmark (UOBM) [6] using the Hermit⁵ hypertableau reasoner. All experiments were performed on a Windows 7 machine with a double core 2.53 GHz Intel x86 64 bit processor and Java 1.6 allowing 1GB of Java heap space. We measure the time for one-off tasks such as classification separately since such tasks are usually performed before the system accepts queries. The ontologies and all code required to perform the experiments are available online.⁶

We first used the 14 conjunctive ABox queries provided in LUBM. From these, queries 2, 7, 8, 9 are the most interesting ones in our setting since they contain many atoms and ordering them can have an effect in running time. We tested the queries on LUBM(1,0) and LUBM(2,0) which contain data for one or two universities respectively, starting from index 0. LUBM(1,0) contains 16,283 individuals and LUBM(2,0) contains 38,334 individuals. LUBM(1,0) took 3.8 s to load and 22.7 s for classification and initialization of known and possible instances of concepts and roles. LUBM(2,0) took 15.8 s to load and 146 s for classification and initialization of known and possible instances. Table 2 shows the execution time for each of the four queries for LUBM(1,0) and LUBM(2,0). The queries marked with (*) are the queries where the static and dynamic algorithms result in the same ordering. In these queries we observe an increase in running time when the dynamic technique is used (in comparison to the static) which is

⁵ <http://www.hermit-reasoner.com/>

⁶ <http://code.google.com/p/query-ordering/>

Table 2. Query answering times in milliseconds for LUBM(1,0) and LUBM(2,0) and in seconds for UOBM (1 university, dep 0-2) using i) the static algorithm ii) the dynamic algorithm and iii) (only for LUBM) 50% sampling

LUMB(1, 0)				LUBM(2,0)			UOBM		
Query	Static	Dynamic	Sampling	Static	Dynamic	Sampling	Query	Static	Dynamic
2	110	203	721	386	917	12,767	4	23.81	24.08
*7	47	66	1,743	138	127	9,653	9	701.24	690.51
*8	686	873	867	2,434	4,237	2,488	11	2.22	2.07
9	1,670	13,056	13,372	17,960	91,787	94,087	12	0.13	0.16
							14	212.28	215.56
							q ₁	668.82	347.25
							q ₂	179.92	85.65

especially evident on LUBM(2,0) Query 8, where the number of individuals in the ontology and the intermediate result sizes are larger. Dynamic ordering also behaves worse than static in queries 2 and 9. This happens because, although the dynamic algorithm chooses a better ordering than the static algorithm, the intermediate results (that need to be checked in each iteration to determine the next query atom to be executed) are quite large and hence the cost of iterating over all possible mappings in the dynamic case far outweighs the better ordering that is obtained. We also observe that a random sampling of individuals for collecting the ordering statistics in the dynamic case (checking only 50% of individuals in Ω_{i-1} randomly for detecting the next query atom to be executed) leads to much worse results in most queries than plain static or dynamic ordering.

From the nondeterministic UOBM ontology we removed the nominals and only used the first three departments containing 6,409 individuals. The ontology took 6.4 s to load and 30.3 s to classify and initialize the known and possible instances. We ran our static and dynamic algorithms on queries 4, 9, 11, 12 and 14 provided in UOBM, which are the most interesting ones because they consist of many atoms. Most of these queries contain one atom with possible instances. Static and dynamic ordering show similar performance because the intermediate result set sizes are small and the available statistics in this case are quite accurate and result in the same ordering for both methods. For both ordering methods, atoms with possible instances for these queries are executed last. In order to illustrate when dynamic ordering performs better than static, we also created the two custom queries:

$$q_1 = \{ \text{isAdvisedBy}(x,y), \text{GraduateStudent}(x), \text{Woman}(y) \}$$

$$q_2 = \{ \text{SportsFan}(x), \text{GraduateStudent}(x), \text{Woman}(x) \}$$

In both queries, $P[\text{GraduateStudent}]$, $P[\text{Woman}]$ and $P[\text{isAdvisedBy}]$ are non-empty. The running times for dynamic ordering are smaller since the more accurate statistics result in a smaller number of possible instances that have to be checked during query execution. In particular, for the static ordering, 151 and 41 possible instances have to be checked in query q_1 and q_2 , respectively, compared to only 77 and 23 for the dynamic ordering. Moreover, the intermediate results are generally smaller in dynamic ordering than in static leading to a significant reduction in the running time of the queries. All of the presented queries could not be answered in the time limit of 30 minutes in case no ordering algorithm was used.

6 Conclusions

In the current paper, we have dealt with the definition of cost formulas that are based on information extracted from reasoners' models for ordering the atoms of a conjunctive instance query that is issued over an OWL ontology. We have devised two algorithms, a static and a dynamic one, for finding a good order and show (through an experimental study) that static techniques are quite adequate for deterministic ontologies, however, when disjunctive knowledge is present, dynamic techniques often perform better. The proposed query ordering costs can be used with either tableau or hypertableau reasoners, however, in the case of tableau reasoners they can be less accurate. Future work will include the definition of additional cost measures and the use of appropriate criteria for sampling the individuals and use only these samples for extracting the costs for dynamic ordering.

Acknowledgements This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

1. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. *Journal of Artificial Intelligence Research* 31, 151–198 (2008)
2. Glimm, B., Horrocks, I., Motik, B., Shearer, R., Stoilos, G.: A novel approach to ontology classification. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Accepted (2012)
3. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. *J. Web Semantics* 3(2-3), 158–182 (2005)
4. Kazakov, Y.: *RIQ* and *SROIQ* are harder than *SHOIQ*. In: Brewka, G., Lang, J. (eds.) *Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'08)*. pp. 274–284. AAAI Press (2008)
5. Kollia, I., Glimm, B., Horrocks, I.: SPARQL query answering over OWL ontologies. In: *Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011)*. pp. 382–396. *Lecture Notes in Computer Science*, Springer-Verlag (2011)
6. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: *The Semantic Web: Research and Applications*, chap. 12, pp. 125–139. *Lecture Notes in Computer Science*, Springer (2006)
7. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research* 36, 165–228 (2009)
8. Sirin, E., Parsia, B.: Optimizations for answering conjunctive ABox queries: First results. In: *Proc. of the Int. Description Logics Workshop DL* (2006)
9. Steinbrunn, M., Moerkotte, G., Kemper, A.: Heuristic and randomized optimization for the join ordering problem. *VLDB Journal* 6, 191–208 (1997)
10. Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL basic graph pattern optimization using selectivity estimation. In: *Proceedings of the 17th international conference on World Wide Web*. pp. 595–604. *WWW '08*, ACM, New York, NY, USA (2008)

Inconsistency-Tolerant First-Order Rewritability of DL-Lite with Identification and Denial Assertions

Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati,
Marco Ruzzi, and Domenico Fabio Savo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti
Sapienza Università di Roma
lembo,lenzerini,rosati,ruzzi,savo@dis.uniroma1.it

Abstract. This paper is motivated by two requirements arising in practical applications of ontology-based data access (OBDA): the need of inconsistency-tolerant semantics, which allow for dealing with classically inconsistent specifications; and the need of expressing assertions which go beyond the expressive abilities of traditional Description Logics, namely identification and denial assertions. We consider an extension of DL-Lite (the most used DL in OBDA) which allows for the presence of both the aforementioned kinds of assertions in the TBox. We adopt an inconsistency-tolerant semantics for such a DL, specifically the so-called Intersection ABox Repair (IAR) semantics, and we study query answering in this setting. Our main contribution is a query rewriting technique which is able to take into account both identification and denial assertions. By virtue of this technique, we prove that conjunctive query answering under such semantics is first-order rewritable in the considered extension of DL-Lite.

1 Introduction

Ontology-based data access (OBDA) is a computing paradigm which considers the problem of accessing data stored in autonomous databases through the use of an ontology, which provides a conceptual and shared representation of the domain of interest [10]. The main components of an OBDA system are the ontology, the set of data sources to be accessed, and the mapping, which specifies the relationship between the ontology and the data sources. The reasoning service of main interest is query answering, which amounts to process a query expressed in terms of the ontology alphabet, and to compute its answer on the basis of the knowledge specified by the ontology, the mapping, and the data stored at the sources. Various languages for specifying the ontology in this context have been recently proposed [3,2,7], which are designed in order to allow for tractable query answering w.r.t. the size of the data (data complexity). Among these, the members of the *DL-Lite* family of light-weight Description Logics (DLs) present the distinguishing feature of *first-order rewritable query answering* for unions of conjunctive queries (UCQs), which means that such a reasoning service can be reduced to the evaluation of a first-order query (directly translatable into SQL) over a database. This turns out to be a crucial aspect in OBDA, where data are in general not moved from the sources, and are usually managed by a Relational Database Management System.

In the last years we have experimented (see, e.g., [11]) that two important requirements arise in OBDA applications: (*i*) the need of declarative mechanisms for dealing

with data that are inconsistent with respect to the intensional knowledge specified by the ontology, and (ii) the need of expressing assertions which go beyond the expressive abilities of traditional DLs, namely *identification assertions* and *denial assertions*. Solutions aiming at fulfilling these two requirements should of course still guarantee enough efficiency. Ideally, they should allow for inconsistency-tolerant first-order rewritable query answering of UCQs. Devising one such solution is the goal of the present paper.

Motivated by the first mentioned requirement, we have recently proposed various *inconsistency-tolerant semantics* [8], inspired by the studies on inconsistency handling in belief revision [6] and by the work on consistent query answering in databases [5]. Later works [9,1] have then shown that answering UCQs under the so-called *Intersection ABox Repair (IAR) semantics* is first-order rewritable for ontologies specified in $DL\text{-}Lite_A$, a member of the $DL\text{-}Lite$ family [9]. In the present paper we extend the above result, and show that first-order rewritability of conjunctive query answering still holds if we enrich $DL\text{-}Lite_A$ with identification and denial assertions.

Identification assertions (IdAs) are mechanisms for specifying a set of properties that can be used to identify instances of concepts. IdAs we study here have been originally presented in [4]. Such assertions allow for sophisticated forms of object identification, which may include paths realized through the chaining of roles, their inverses, and attributes. Roughly speaking, when we say that instances of a concept C are identified by a path π , we impose that no two instances of C with overlapping sets of π -fillers exist, i.e., in every interpretation \mathcal{I} , no two instances o and o' of C exist with a non-empty intersection of the set of objects reachable by o and by o' in \mathcal{I} . This naturally extends to IdAs involving more than one path. Identification assertions are very useful in practice and are essential to represent n -relations and attributes of roles through reification. Indeed, reification is the only way to model n -relations and role attributes in ontology languages, such as OWL, which do not natively allow for such constructs.

Denial Assertions (DAs) are instead used to impose that the answer to a certain boolean conjunctive query over the ontology is false, analogous to negative constraints in [2]. This is particularly useful to specify general forms of disjointness, which is again not supported in traditional ontology languages.

The query rewriting algorithm given in this paper elegantly deals with all forms of inconsistency that can arise in $DL\text{-}Lite_A$ enriched with IdAs and DAs, thus it represents an alternative to the rewriting algorithm given in [9] for $DL\text{-}Lite_A$ only. Moreover, it properly manages all the inconsistencies caused by erroneous assignments of values to attributes, i.e., arising when a value of type T_i is assigned to an attribute of value-type T_j disjoint from T_i . This kind of inconsistency has not been considered in [9].

In this paper, for the sake of simplicity, we consider the setting of a single ontology constituted by a TBox and an ABox. However, our technique can be easily extended to the OBDA setting, where mapping assertions relate the TBox to the data sources [10].

The paper is organized as follows. In section 2 we provide some preliminaries. In Section 3 we provide some general properties of the *IAR*-semantics, useful for the rest of the paper. In Section 4, we show first-order rewritability of query answering of UCQs under the *IAR*-semantics. In Section 5 we conclude the paper.

2 Preliminaries

Let Σ be a signature partitioned into Σ_P , containing symbols for predicates, i.e., atomic concepts, atomic roles, attributes and value-domains, and Σ_C , containing symbols for individual (object and value) constants. Given a DL language \mathcal{L} , an \mathcal{L} -ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ over Σ consists of a TBox \mathcal{T} , i.e., a set of intensional assertions over Σ expressed in \mathcal{L} , and an ABox \mathcal{A} , i.e., a set of extensional assertions over Σ expressed in \mathcal{L} . We assume that ABox assertions are always atomic, i.e., they correspond to *ground atoms*, and therefore we omit to refer to \mathcal{L} when we talk about ABox assertions.

The semantics of a DL ontology \mathcal{O} is given in terms of first-order (FOL) interpretations. We denote with $Mod(\mathcal{O})$ the set of models of \mathcal{O} , i.e., the set of FOL-interpretations that satisfy all the assertions in \mathcal{T} and \mathcal{A} , where the definition of satisfaction depends on the DL language in which \mathcal{O} is specified. An ontology \mathcal{O} is *satisfiable* if $Mod(\mathcal{O}) \neq \emptyset$, and \mathcal{O} *entails* a FOL-sentence ϕ , denoted $\mathcal{O} \models \phi$, if ϕ is satisfied by every $\mathcal{I} \in Mod(\mathcal{O})$. The above definitions naturally apply to a TBox alone.

In the following, we consider DL ontologies $\langle \mathcal{T}, \mathcal{A} \rangle$ in which the TBox \mathcal{T} is always satisfiable, whereas the ABox \mathcal{A} may contradict \mathcal{T} . When this happens, we say that \mathcal{A} is *\mathcal{T} -inconsistent*, \mathcal{T} -consistent otherwise. For ontologies of this kind, the *Intersection ABox Repair (IAR)* semantics introduced in [8] allows for meaningful reasoning in the presence of inconsistency, which is instead trivialized under the FOL-semantics. The *IAR*-semantics is defined in terms of *A-repairs*: given an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, an *A-repair* for \mathcal{O} is a set $\mathcal{A}' \subseteq \mathcal{A}$ such that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$, and there does not exist \mathcal{A}'' such that $\mathcal{A}' \subset \mathcal{A}'' \subseteq \mathcal{A}$ and $Mod(\langle \mathcal{T}, \mathcal{A}'' \rangle) \neq \emptyset$. In other terms, an *A-repair* for \mathcal{O} is a maximal \mathcal{T} -consistent subset of \mathcal{A} . The set of *A-repairs* for \mathcal{O} is denoted by $AR-Set(\mathcal{O})$. The *IAR*-semantics is then given in terms of *IAR*-models, as follows.

Definition 1. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL ontology. The set of Intersection ABox Repair (IAR)-models of \mathcal{O} is defined as $Mod_{IAR}(\mathcal{O}) = Mod(\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR-Set(\mathcal{O})} \mathcal{A}_i \rangle)$*

Notice that if \mathcal{O} is satisfiable under FOL-semantics, then $Mod(\mathcal{O}) = Mod_{IAR}(\mathcal{O})$. The notion of consistent entailment is then the natural extension of classical entailment to *IAR*-semantics: a FOL-sentence ϕ is *IAR-consistently entailed*, or simply *IAR-entailed*, by \mathcal{O} , written $\mathcal{O} \models_{IAR} \phi$, if $\mathcal{I} \models \phi$ for every $\mathcal{I} \in Mod_{IAR}(\mathcal{O})$.

We focus now on *DL-Lite_{A,id}*, a DL of the *DL-Lite* family [3] equipped with identification assertions [4]. Since *DL-Lite_{A,id}* distinguishes between object and value constants, we partition the set Σ_C into two disjoint sets Σ_O , which is the set of constants denoting objects, and Σ_V , which is the set of constants denoting values.

Basic *DL-Lite_{A,id}* expressions are defined as follows:

$$\begin{array}{lll}
 B & \longrightarrow & A \mid \exists Q \mid \delta(U) \\
 C & \longrightarrow & B \mid \neg B \\
 Q & \longrightarrow & P \mid P^- \\
 R & \longrightarrow & Q \mid \neg Q \\
 E & \longrightarrow & \rho(U) \\
 F & \longrightarrow & T_1 \mid \dots \mid T_n \\
 V & \longrightarrow & U \mid \neg U
 \end{array}$$

A , P , and P^- denote an *atomic concept*, an *atomic role*, and the *inverse of an atomic role*; $\delta(U)$ (resp. $\rho(U)$) denotes the *domain* (resp. the *range*) of an *attribute* U , i.e., the set of objects (resp. values) that U relates to values (resp. objects); T_1, \dots, T_n are unbounded pairwise disjoint predefined value-domains; B is called *basic concept*.

A $DL\text{-Lite}_{A,id}$ TBox is a finite set of the following assertions:

$$B \sqsubseteq C \quad Q \sqsubseteq R \quad U \sqsubseteq V \quad E \sqsubseteq F \quad (\text{funct } Q) \quad (\text{funct } U) \quad (\text{id } B \pi_1, \dots, \pi_n)$$

The assertions above, from left to right, respectively denote inclusions between concepts, roles, attributes, and value-domains, global functionality on roles and on attributes, and identification assertions (IdAs). In IdAs, π_i is a *path*, which is an expression built according to the following syntax: $\pi \rightarrow S \mid D? \mid \pi_1 \circ \pi_2$, where S denotes an atomic role, the inverse of an atomic role, an attribute, or the inverse of an attribute, $\pi_1 \circ \pi_2$ denotes the composition of paths π_1 and π_2 , and $D?$, called *test relation*, represents the identity relation on instances of D , which can be a basic concept or a value-domain. Test relations are used to impose that a path involves instances of a certain concept or value-domain. In our logic, IdAs are *local*, i.e., at least one $\pi_i \in \{\pi_1, \dots, \pi_n\}$ has length 1, i.e., it is an atomic role, the inverse of an atomic role, or an attribute. Intuitively, an IdA of the above form asserts that for any two different instances o, o' of B , there is at least one π_i such that o and o' differ in the set of their π_i -fillers, that is the set of objects that are reachable from o by means of π_i . For example, the IdA $(\text{id } Match \text{ homeTeam}, \text{visitorTeam})$ says that there are not two different matches with the same home team and host team (which is indeed what happens, for instance, in a season schedule of a football league).

Inclusion assertions that do not contain (resp. contain) the symbols ' \neg ' in the right-hand side are called *positive inclusions* (resp. *negative inclusions*). The set of positive (resp., negative) inclusions in \mathcal{T} will be denoted by \mathcal{T}^+ (resp., \mathcal{T}^-).

A $DL\text{-Lite}_{A,id}$ ABox is a finite set of assertions of the form $A(a)$, $P(a, b)$, and $U(a, v)$, where A is an atomic concept, P is an atomic role, U is an attribute, a and b are constants of Σ_O , and v is a constant of Σ_V . In a $DL\text{-Lite}_{A,id}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, the following condition holds: each role or attribute that either is functional in \mathcal{T} or appears (in either direct or inverse direction) in a path of an IdA in \mathcal{T} is not specialized, i.e., it does not appear in the right-hand side of assertions of the form $Q \sqsubseteq Q'$ or $U \sqsubseteq U'$.

The semantics of $DL\text{-Lite}_{A,id}$ is given in [4]. We only recall here that each value-domain T_i is interpreted by means of the same function, denoted $val(T_i)$, in every interpretation, and each constant $c_i \in \Sigma_V$ is interpreted as one specific value, denoted $val(c_i)$. Also, the Unique Name Assumption is adopted.

A *boolean conjunctive query* (BCQ) over a $DL\text{-Lite}_A$ ontology is an expression of the form $q = \exists \vec{y}.conj(\vec{t})$ where \vec{y} is a set of variables and \vec{t} is a set of terms (i.e., constants or variables) such that each variable in \vec{t} is also in \vec{y} , and $conj(\vec{t})$ is a conjunction of atoms of the form $A(z)$, $T(z')$, $P(z, z')$ $U(z, z')$ where A is an atomic concept, T is a value-domain, P is an atomic role and U is an attribute, and z, z' are terms. In the following, we will mainly consider *boolean unions of conjunctive queries* (BUCQs), i.e., first order sentences of the form $Q = \bigvee_{i=1}^n q_i$ where $q_i = \exists \vec{y}_i.conj_i(\vec{t}_i)$ is a BCQ.

A BCQ q is satisfied by an ABox \mathcal{A} (denoted by $\langle \emptyset, \mathcal{A} \rangle \models q$) if there exists a substitution σ from the variables in q to constants of Σ_C such that the formula $\sigma(q)$ is satisfied in the FOL-interpretation structure where the ABox \mathcal{A} determines the extension of concepts, roles and attributes and the function val determines the extension of the value-domains. With a little abuse of notation we sometimes use $\sigma(q)$ to indicate the set of the atoms in $\sigma(q)$. When query q is satisfied by \mathcal{A} , we say that $\sigma_{\mathcal{A}}(q) = \sigma(q) \cap \mathcal{A}$

is the *image of q in \mathcal{A}* . A BUCQ $Q = \bigvee_{i=1}^n q_i$ is satisfied by an ABox \mathcal{A} if there exists $i \in \{1, \dots, n\}$ such that q_i is satisfied by \mathcal{A} .

All the results of the present work can be straightforwardly extended to *non-boolean* UCQs, by imposing the (natural) condition that every free variable in the query appears in at least one atom not involving value-domains.

With the notion of query in place we introduce now *denial assertions (DAs)*. A DA is an expression of the form $\forall \vec{y}. \text{conj}(\vec{t}) \rightarrow \perp$ where $\text{conj}(\vec{t})$ is defined as for BCQs. A DA is satisfied by an interpretation \mathcal{I} if the formula $\exists \vec{y}. \text{conj}(\vec{t})$ evaluates to `false` in \mathcal{I} . DAs are used to impose general forms of disjointness. For example, the denial $\forall x, y. \text{Match}(x) \wedge \text{homeTeam}(x, y) \wedge \text{visitorTeam}(x, y) \rightarrow \perp$ says that a match cannot have the same team as both home and visitor team.

In the following we will consider $DL\text{-Lite}_{A, id}$ ontologies whose TBoxes are enriched with DAs, and call them $DL\text{-Lite}_{A, id, den}$ ontologies.

3 Properties of the IAR-Semantics

We discuss here some properties of the *IAR*-semantics that will be used in the rest of the paper. We recall that the *IAR*-semantics is defined for DL ontologies composed by a consistent TBox and an ABox comprising only atomic assertions, therefore we consider below only this kind of ontologies. We start with the notion of inconsistent set.

Definition 2. Let \mathcal{T} be a TBox and let \mathcal{A} be an ABox. \mathcal{A} is a minimal \mathcal{T} -inconsistent set if \mathcal{A} is \mathcal{T} -inconsistent, i.e., the ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable, and there is no $\mathcal{A}' \subset \mathcal{A}$ such that \mathcal{A}' is \mathcal{T} -inconsistent.

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL ontology. We denote with $\text{minIncSets}(\mathcal{O})$ the set of minimal \mathcal{T} -inconsistent sets contained in \mathcal{A} . Notice that \mathcal{O} is satisfiable iff $\text{minIncSets}(\mathcal{O}) = \emptyset$. Then, the following property holds.

Proposition 1. Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL ontology such that $\text{minIncSets}(\mathcal{O}) \neq \emptyset$, and let α be an ABox assertion in \mathcal{A} . An \mathcal{A} -repair \mathcal{A}' of \mathcal{O} such that $\alpha \notin \mathcal{A}'$ exists iff there exists $V \in \text{minIncSets}(\mathcal{O})$ such that $\alpha \in V$.

Let now $\text{AR-Set}(\mathcal{O}) = \{\mathcal{A}_1, \dots, \mathcal{A}_n\}$ be the set of \mathcal{A} -repairs of an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, and let q be a BCQ. From Definition 1, we derive that $\mathcal{O} \models_{IAR} q$ iff there exists a subset \mathcal{A}' of \mathcal{A} such that $\mathcal{A}' \subseteq \mathcal{A}_i$ for every $1 \leq i \leq n$ and $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$. From the observation above, we derive the following theorem, which characterizes the notion of *IAR*-entailment of BCQs.

Theorem 1. Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL ontology such that $\text{minIncSets}(\mathcal{O}) \neq \emptyset$, and let q be a BCQ. $\mathcal{O} \models_{IAR} q$ iff there exists $\mathcal{A}' \subseteq \mathcal{A}$ such that: (i) \mathcal{A}' is \mathcal{T} -consistent; (ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$; (iii) $\mathcal{A}' \cap V = \emptyset$ for every $V \in \text{minIncSets}(\mathcal{O})$.

4 Query Rewriting under IAR-Semantics in $DL\text{-Lite}_{A, id, den}$

In this section, we focus on $DL\text{-Lite}_{A, id, den}$ ontologies, and provide our technique for computing FOL-rewritings of BUCQs under the *IAR*-semantics. The notion of FOL-rewritability is essentially the same used under FOL-semantics [3]. More precisely,

BUCQs in $DL-Lite_{A,id,den}$ are FOL-rewritable under the IAR -semantics if, for each BUCQs q and each $DL-Lite_{A,id,den}$ TBox \mathcal{T} , there exists a FOL-query q_r such that, for any ABox \mathcal{A} $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} q$ iff $\langle \emptyset, \mathcal{A} \rangle \models q_r$. Such q_r is called the *IAR-perfect reformulation* of q w.r.t. \mathcal{T} .

To come up with our reformulation method, we exploit Theorem 1. Roughly speaking, in the reformulation of a BUCQ q over a $DL-Lite_{A,id,den}$ TBox \mathcal{T} , we encode into a FOL-formula all violations that can involve assertions belonging to images of q in any ABox \mathcal{A} . Indeed, this can be done by reasoning only on the TBox, and considering each query atom separately. Intuitively, we deal with inconsistency by rewriting each atom α of q into a FOL-formula α_r in such a way that $\langle \emptyset, \mathcal{A} \rangle \models \alpha_r$ only if there exists a substitution σ of the variables in q to constants of Σ_C such that q is satisfied by \mathcal{A} , and $\sigma(\alpha)$ does not belong to any minimal \mathcal{T} -inconsistent set.

This inconsistency-driven rewriting is then suitably casted into the final reformulation, which takes into account also positive knowledge of the TBox, i.e., the inclusion assertions in \mathcal{T}^+ . As we will show later in this section, this can be done by means of a slight variation of the algorithm **PerfectRef** of [3,10]. To formalize the above idea, we need to introduce some preliminary definitions.

We consider Boolean queries corresponding to FOL-sentences of the form:

$$\exists z_1, \dots, z_k. \bigwedge_{i=1}^n H_i(t_i^1) \wedge \bigwedge_{i=1}^m \neg T_i(t_i^2) \wedge \bigwedge_{i=1}^{\ell} S_i(t_i^3, t_i^4) \wedge \bigwedge_{i=1}^h t_i^5 \neq t_i^6 \quad (1)$$

where every H_i is an unary predicate, which is either an atomic concept or a value-domain, every T_i is a value-domain, every S_i is a binary predicate, which is either an atomic role or an attribute, every t_i^j is a term (i.e., either a constant or a variable), and z_1, \dots, z_k are all the variables of the query. Notice that sentences of the form (1) are BCQs enriched with limited forms of inequalities and negation.

Given a $DL-Lite_{A,id,den}$ TBox \mathcal{T} , we denote with $contr(\mathcal{T})$ the set of all negative inclusions, functionalities, identifications, denials, and value-domain inclusions occurring in \mathcal{T} . Intuitively, the set $contr(\mathcal{T})$ contains all those assertions in \mathcal{T} which can be contradicted by assertions in the ABox. Now, to each assertion $\tau \in contr(\mathcal{T})$, we associate a Boolean query, denoted $\varphi(\tau)$, which encodes the violation of τ , i.e., it looks for images of the negation of τ . Below we provide some of such encodings. Missing cases are can be obtained in a similar way.

$$\begin{aligned} -\varphi(\text{(funct } P)) & : \exists x, x_1, x_2. P(x, x_1) \wedge P(x, x_2) \wedge x_1 \neq x_2 \\ -\varphi(A_1 \sqsubseteq \neg A_2) & : \exists x. A_1(x) \wedge A_2(x) \\ -\varphi(\rho(U) \sqsubseteq T) & : \exists x_1, x_2. U(x_1, x_2) \wedge \neg T(x_2) \\ -\varphi(\forall \vec{x}. conj(\vec{t}) \rightarrow \perp) & : \exists \vec{x}. conj(\vec{t}) \end{aligned}$$

For example, $\varphi(\text{(id } Match \ homeTeam, visitorTeam)) = \exists x, x', y, z. Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, z) \wedge Match(x') \wedge homeTeam(x', y) \wedge visitorTeam(x', z) \wedge x \neq x'$, and $\varphi(\forall x, y. Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, y) \rightarrow \perp) = \exists x, y. Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, y)$.

Then, the algorithm **unsatQueries**(\mathcal{T}) computes the first-order reformulation under FOL-semantics of $\varphi(\tau)$ for each $\tau \in contr(\mathcal{T})$, and returns the union of all such reformulations. To this aim, **unsatQueries**(\mathcal{T}) makes use of the algorithm

$\text{PerfectRef}_{\neq}(\mathcal{T}^+, \varphi(\tau))$, which is a slight variation of the algorithm PerfectRef given in [3]. In this modified version, inequality is considered as a primitive role and negated value-domains are considered as primitive concepts, thus inequality and negated atoms are never rewritten by the algorithm, and the algorithm does not unify query atoms if this causes a violation of an inequality. Note that the result of PerfectRef_{\neq} is a union of boolean queries of the form (1), represented as a set of queries, as usual.

For example, assume to have a TBox containing the above mentioned IdA $\tau_1 : (\text{id Match homeTeam, visitorTeam})$ and the inclusion assertion $\text{playedMatch} \sqsubseteq \text{Match}$. The algorithm $\text{PerfectRef}_{\neq}(\mathcal{T}^+, \varphi(\tau_1))$ returns, among others, the query:

$$\exists x, x', y, z. \text{playedMatch}(x) \wedge \text{homeTeam}(x, y) \wedge \text{visitorTeam}(x, z) \wedge \text{playedMatch}(x') \wedge \text{homeTeam}(x', y) \wedge \text{visitorTeam}(x', z) \wedge x \neq x'$$

Notice that the query $\varphi(\tau_1)$ cannot be rewritten by unifying $\text{Match}(x)$ and $\text{Match}(x')$ because of the inequality $x \neq x'$. Such an inequality actually causes that in this example the algorithm can never rewrite queries through unification.

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $DL\text{-Lite}_{A, id, den}$ ontology. Since \mathcal{O} is satisfiable iff there are no \mathcal{T} -inconsistent sets in \mathcal{A} , the following theorem states that the query produced by the algorithm $\text{unsatQueries}(\mathcal{T})$ can be used to check satisfiability of \mathcal{O} .

Theorem 2. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $DL\text{-Lite}_{A, id, den}$ ontology. A \mathcal{T} -inconsistent set $V \subseteq \mathcal{A}$ exists iff $\langle \emptyset, \mathcal{A} \rangle \models \text{unsatQueries}(\mathcal{T})$.*

If $\langle \emptyset, \mathcal{A} \rangle \models \text{unsatQueries}(\mathcal{T})$, then there exists $q \in \text{unsatQueries}(\mathcal{T})$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. This implies that there exists a substitution σ from the variables in q to constants of Σ_C such that $\sigma(q)$ projected over its positive atoms is contained in \mathcal{A} . Similar to BCQs, we call this set the image of q in \mathcal{A} , denoted $\sigma_{\mathcal{A}}(q)$. It is possible to show that $\sigma_{\mathcal{A}}(q)$ is a \mathcal{T} -inconsistent set contained in \mathcal{A} (in fact, this is part of the proof of Theorem 2). In order to exploit Theorem 1 towards the definition of a FOL-rewriting procedure, we need however to identify those assertions in \mathcal{A} that participate to a *minimal* \mathcal{T} -inconsistent set. From the definition of minimal \mathcal{T} -inconsistent set, and from Theorem 2 it follows that $\sigma_{\mathcal{A}}(q)$ is a minimal \mathcal{T} -inconsistent set iff for every $V \subset \sigma_{\mathcal{A}}(q)$ and every query $q' \in \text{unsatQueries}(\mathcal{T})$, we have that $\langle \emptyset, V \rangle \not\models q'$.

Based on the above observations, we introduce the algorithm $\text{minUnsatQueries}(\mathcal{T})$ which, starting from the set $\text{unsatQueries}(\mathcal{T})$, computes a new set \mathcal{Q}_{min} of queries, which enjoys the following properties: (i) For each query $q \in \mathcal{Q}_{min}$, $\langle \emptyset, \mathcal{A} \rangle \models q$ iff there exists in $\text{unsatQueries}(\mathcal{T})$ a query q' such that $\langle \emptyset, \mathcal{A} \rangle \models q'$. This guarantees that Theorem 2 also holds with $\text{minUnsatQueries}(\mathcal{T})$ in place of $\text{unsatQueries}(\mathcal{T})$. (ii) For each query $q \in \mathcal{Q}_{min}$, if $\langle \emptyset, \mathcal{A} \rangle \models q$, then for every image $\sigma_{\mathcal{A}}(q)$ of q in \mathcal{A} , $\langle \emptyset, V \rangle \not\models q'$, where $V \subset \sigma_{\mathcal{A}}(q)$ and $q' \in \mathcal{Q}_{min}$. This guarantees that if a query $q \in \mathcal{Q}_{min}$ is such that $\langle \emptyset, \mathcal{A} \rangle \models q$, then every image of q in \mathcal{A} is a minimal \mathcal{T} -inconsistent set.

Before presenting the algorithm $\text{minUnsatQueries}(\mathcal{T})$, we need to introduce some preliminary notions. Given a query q , we say that a term t occurs in an *object position* of q if q contains an atom of the form $A(t)$, $P(t, t')$, $P(t', t)$, $U(t, t')$, whereas we say that t occurs in a *value position* of q if q contains an atom of the form $T(t)$, $U(t', t)$, where A , P , U , and T have the usual meaning. Given two terms t_1 and t_2 occurring in a query q , we say that t_1 and t_2 are *compatible* in q if either both t_1 and t_2 appear only

in object positions of q or both t_1 and t_2 appear only in value positions of q . Moreover, let q and q' be two boolean queries. We say that q is a *proper syntactical subset* of q' , written $q \prec_{Rn} q'$ if there exists a renaming function $Rn(q, q')$ of the variables in q to the variables in q' , such that every atom $S(\vec{t})$ occurring in $Rn(q, q')$ occurs also in q' and an analogous renaming from q' to q does not exist. The algorithm $\text{minUnsatQueries}(\mathcal{T})$ is given below.

Algorithm: $\text{minUnsatQueries}(\mathcal{T})$
Input: a $DL\text{-Lite}_{A,id,den}$ TBox \mathcal{T}
Output: a set of queries

```

1  $\mathcal{Q}' \leftarrow \text{unsatQueries}(\mathcal{T});$ 
2  $\mathcal{Q}'' \leftarrow \text{saturate}(\mathcal{Q}')$ ;
3  $\mathcal{Q}''' \leftarrow \emptyset;$ 
4 while  $\mathcal{Q}''' \neq \mathcal{Q}''$  do
5    $\mathcal{Q}''' \leftarrow \mathcal{Q}'';$ 
6   foreach  $q \in \mathcal{Q}''$  do
7     foreach atom  $U(t, t')$  in  $q$  do
8       if there exist no atoms  $T(t')$  and  $\neg T(t')$  in  $q$  then
9         foreach value-domain  $T$  in  $\{T_1, \dots, T_n\}$  do
10           $\mathcal{Q}'' \leftarrow \mathcal{Q}'' \setminus \{q\};$ 
11           $\mathcal{Q}'' \leftarrow \mathcal{Q}'' \cup \{q \wedge T(t')\};$ 
12           $\mathcal{Q}'' \leftarrow \mathcal{Q}'' \cup \{q \wedge \neg T(t')\};$ 
13 foreach  $q \in \mathcal{Q}'''$  do
14   foreach term  $t$  occurring in  $q$  do
15     if both  $T_i(t)$  and  $T_j(t)$  occur in  $q$ , with  $i \neq j$  then  $\mathcal{Q}''' \leftarrow \mathcal{Q}''' \setminus \{q\};$ 
16 foreach  $q$  and  $q'$  in  $\mathcal{Q}'''$  do
17   if  $q \prec_{Rn} q'$  then  $\mathcal{Q}''' \leftarrow \mathcal{Q}''' \setminus \{q'\};$ 
18 return  $\mathcal{Q}'''$ 

```

The algorithm proceeds as follows.

Step 1 (line 1): the algorithm initializes \mathcal{Q}' to the set $\text{unsatQueries}(\mathcal{T})$.

Step 2 (line 2): the algorithm computes the set \mathcal{Q}'' through the algorithm saturate . Starting from each query $q \in \mathcal{Q}'$, such an algorithm first unifies pairs of compatible terms in q in all possible ways; then, for any query q' computed in this way, for each pair of terms t_1 and t_2 occurring in q' that are syntactically different, it adds the inequality atom $t_1 \neq t_2$ to q' ; finally it returns the union of \mathcal{Q}' with this new set of queries. Notice that such an operation is sound and complete with respect to the set of queries in \mathcal{Q}' : in particular, no answer to the set of queries is lost by this transformation, since, for every pair of *compatible* terms t_1, t_2 which are forced to be not equal in q , there is a query q' in \mathcal{Q}' where the same terms have been unified. For instance, assume that \mathcal{Q}' contains only the query $q : \exists x, y. \text{Match}(x) \wedge \text{homeTeam}(x, y) \wedge \text{visitorTeam}(x, y)$. Then, the algorithm saturate returns a set constituted by the queries $q_1 : \exists x, y. \text{Match}(x) \wedge \text{homeTeam}(x, y) \wedge \text{visitorTeam}(x, y) \wedge x \neq y$ and $q_2 : \exists x. \text{Match}(x) \wedge \text{homeTeam}(x, x) \wedge \text{visitorTeam}(x, x)$.

Step 3 (lines 3-12): let $\{T_1 \dots T_n\}$ be the set of value-domains. The algorithm computes the set \mathcal{Q}''' by producing from each query $q \in \mathcal{Q}''$ the following queries: for each atom $U(x, y)$, where U is an attribute name, if no atoms $T(y)$ appears in q , where T is a value-domain, then the algorithm builds $2n$ new queries by substituting the atom $U(x, y)$ with either the conjunction of atoms $U(x, y) \wedge T_i(y)$ or $U(x, y) \wedge \neg T_i(y)$, for

each $1 \leq i \leq n$. In this way, every possible combination is computed. It is easy to verify that such a transformation is also sound and complete. Step 3 is motivated by the need to provide a complete comparison in Step 5, as explained below.

Step 4 (lines 13-15): the algorithm removes from \mathcal{Q}''' every query q in which a term t occurs in two atoms of the form $T_1(t)$ and $T_2(t)$ (T_1 and T_2 are disjoint, and therefore for each constant c in Σ_V , $T_1(c) \wedge T_2(c)$ is a contradiction).

Step 5 (lines 16-17): the algorithm removes from the set \mathcal{Q}''' each query q' such that there is in \mathcal{Q}''' a different query q whose atoms form, up to renaming of the variables in q , a proper subset of the atoms appearing in q' . This simplified form of query containment guarantees that for every ABox \mathcal{A} and every query q in \mathcal{Q}''' , there does not exist a query q' in \mathcal{Q}''' , such that for every image $\sigma(q)$ of q in \mathcal{A} , $\langle \emptyset, V \rangle \models q'$ where $V \subset \sigma(q)$.

Step 6 (line 18): the algorithm terminates by returning the set \mathcal{Q}''' .

Let us now continue the example given at Step 2. Assume that \mathcal{Q}' contains also the query $q_3 : \exists x.homeTeam(x, x)$ which is associate to the denial assertion $\forall x.homeTeam(x, x) \rightarrow \perp$. Obviously, besides query q_1 and q_2 , \mathcal{Q}'' contains also q_3 , which is natively in a “saturated” form. Step 3 and Step 4 have no effect in this example, since none of the queries in \mathcal{Q}'' has atoms with attributes or value-domains as predicate. Therefore, $\mathcal{Q}''' = \mathcal{Q}''$, and Step 5 eliminates query q_2 from \mathcal{Q}''' , since $q_3 \prec_{Rn} q_2$. Notice that this step is crucial to ensure that every image of a query in \mathcal{Q}''' in any ABox \mathcal{A} is a *minimal* \mathcal{T} -inconsistent set. Indeed, let us, for instance, pose $\mathcal{A} = \{Match(a), homeTeam(a, a), visitorTeam(a, a)\}$; q_2 has an image in \mathcal{A} , which is \mathcal{A} itself, which is also a \mathcal{T} -inconsistent set, but not minimal: indeed, the set $\{homeTeam(a, a)\} \subset \mathcal{A}$ is also a \mathcal{T} -inconsistent set, since it violates the DA $\forall x.homeTeam(x, x) \rightarrow \perp$. This set is also minimal, and is indeed an image in \mathcal{A} of the query q_3 .

The following lemma states that the algorithm $minUnsatQueries(\mathcal{T})$ can be used to check if a $DL-Lite_{A,id,den}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent.

Lemma 1. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent $DL-Lite_{A,id,den}$ ontology. Then, $\langle \emptyset, \mathcal{A} \rangle \models minUnsatQueries(\mathcal{T})$ iff $\langle \emptyset, \mathcal{A} \rangle \models unsatQueries(\mathcal{T})$.*

The following crucial lemma guarantees instead that one can use the queries produced by the algorithm $minUnsatQueries(\mathcal{T})$ in order to compute every minimal \mathcal{T} -inconsistent set in \mathcal{A} .

Lemma 2. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent $DL-Lite_{A,id,den}$ ontology, and let q be a query in $minUnsatQueries(\mathcal{T})$. If $\langle \emptyset, \mathcal{A} \rangle \models q$, then every image of q in \mathcal{A} is a minimal \mathcal{T} -inconsistent set.*

Let α, β be two atoms. We say that β is compatible with α if there exists a mapping μ of the variables occurring in β to the terms occurring in α such that $\mu(\beta) = \alpha$ (and in this case we denote the above mapping μ with the symbol $\mu_{\alpha/\beta}$). Given an atom α and a query q , we denote by $CompSet(\alpha, q)$ the set of atoms of q which are compatible with α . Then, let \mathcal{T} be a $DL-Lite_{A,id,den}$ TBox and let α be an atom, we define $MinIncSet^{\mathcal{T}}(\alpha)$ as follows.

$$MinIncSet^{\mathcal{T}}(\alpha) = \bigvee_{q \in minUnsatQueries(\mathcal{T}) \wedge CompSet(\alpha, q) \neq \emptyset} \left(\bigvee_{\beta \in CompSet(\alpha, q)} \mu_{\alpha/\beta}(q) \right)$$

The following key property holds.

Theorem 3. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent $DL\text{-Lite}_{A,id,den}$ ontology, and let α be an ABox assertion. There exists a minimal \mathcal{T} -inconsistent set V in \mathcal{A} such that $\alpha \in V$ iff $\langle \emptyset, \mathcal{A} \rangle \models \text{MinIncSet}^{\mathcal{T}}(\alpha)$.*

Let \mathcal{T} be a $DL\text{-Lite}_{A,id,den}$ TBox and let q be a BCQ of the form

$$\exists z_1, \dots, z_k. \bigwedge_{i=1}^n A_i(t_i^1) \wedge \bigwedge_{i=1}^m P_i(t_i^2, t_i^3) \wedge \bigwedge_{i=1}^{\ell} U_i(t_i^4, t_i^5) \quad (2)$$

where all symbols have the usual meaning. We denote by $\text{IncRewr}_{IAR}(q, \mathcal{T})$ the following FOL-sentence:

$$\begin{aligned} \text{IncRewr}_{IAR}(q, \mathcal{T}) = & \exists z_1, \dots, z_k. \bigwedge_{i=1}^n A_i(t_i^1) \wedge \neg \text{MinIncSet}^{\mathcal{T}}(A_i(t_i^1)) \wedge \\ & \bigwedge_{i=1}^m P_i(t_i^2, t_i^3) \wedge \neg \text{MinIncSet}^{\mathcal{T}}(P_i(t_i^2, t_i^3)) \wedge \bigwedge_{i=1}^{\ell} U_i(t_i^4, t_i^5) \wedge \neg \text{MinIncSet}^{\mathcal{T}}(U_i(t_i^4, t_i^5)) \end{aligned}$$

Let \mathcal{Q} be a set of CQs, we define $\text{IncRewrUCQ}_{IAR}(\mathcal{Q}, \mathcal{T}) = \bigvee_{q_i \in \mathcal{Q}} \text{IncRewr}_{IAR}(q_i, \mathcal{T})$. We are then able to give our final results on reformulation of UCQs under the IAR -semantics.

Theorem 4. *Let \mathcal{T} be a $DL\text{-Lite}_{A,id,den}$ TBox and let Q be a UCQ. For every ABox \mathcal{A} , $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} Q$ iff $\langle \emptyset, \mathcal{A} \rangle \models \text{IncRewrUCQ}_{IAR}(\text{saturate}(\text{PerfectRef}(\mathcal{T}^+, Q)), \mathcal{T})$.*

In the above theorem, PerfectRef coincides with the algorithm of [10]. This algorithm takes as input the set of positive inclusions \mathcal{T}^+ and the query Q , and computes the perfect reformulation under FOL-semantics of Q w.r.t. \mathcal{T}^+ . $\text{PerfectRef}(\mathcal{T}^+, Q)$ returns a set of CQs specified over \mathcal{T} . Through this reformulation we first preprocess each query according to “positive” knowledge of the TBox, and then manage it to deal with possible inconsistency. Then, the algorithm saturate previously described is applied to the query thus obtained: this step is necessary for technical reasons (roughly speaking, it is needed in order to exactly identify, for every query atom α , the queries from $\text{minUnsatQueries}(\mathcal{T})$ which correspond to inconsistent sets to which an image of α might belong). This query is finally reformulated according to the definition of IncRewrUCQ_{IAR} .

The following complexity result is a direct consequence of Theorem 4, since establishing whether $\langle \emptyset, \mathcal{A} \rangle \models \text{IncRewrUCQ}_{IAR}(\text{saturate}(\text{PerfectRef}(\mathcal{T}^+, Q)), \mathcal{T})$ simply amounts to evaluating a FOL-query over the ABox \mathcal{A} , which is in AC^0 in data complexity.

Corollary 1. *Let \mathcal{O} be a $DL\text{-Lite}_{A,id,den}$ ontology and let Q be a UCQ. Deciding whether $\mathcal{O} \models_{IAR} Q$ is in AC^0 in data complexity.*

We finally notice that the above results directly imply that query answering of UCQs in $DL\text{-Lite}_{A,id,den}$ under standard semantics is FOL-rewritable, and therefore in AC^0 in data complexity, i.e., it has the same computational behavior of all DLs of the $DL\text{-Lite}$ family.

5 Conclusion

Motivated by the requirements arising in applications of ontology-based data access, we have presented a new algorithm for inconsistency-tolerant query answering in a DL obtained by extending $DL-Lite_A$ with identification and denial assertions. The algorithm is based on a rewriting technique, and shows that query answering under the considered inconsistency-tolerant semantics in $DL-Lite$ remains first-order rewritable, even when identification and denial assertions are added to the TBox. We will soon experiment our new technique in the context of several ongoing OBDA projects. Our main goal is to devise optimization techniques that allow for simplifying the rewritten query as much as possible, so that the performance of the query answering process remains acceptable even under the inconsistency-tolerant semantics.

References

1. Biennu, M.: First-order expressibility results for queries over inconsistent $DL-Lite$ knowledge bases. In: Proc. of DL 2011. CEUR, ceur-ws.org, vol. 745 (2011)
2. Cali, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of PODS 2009. pp. 77–86 (2009)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The $DL-Lite$ family. J. of Automated Reasoning 39(3), 385–429 (2007)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Path-based identification constraints in description logics. In: Proc. of KR 2008. pp. 231–241 (2008)
5. Chomicki, J.: Consistent query answering: Five easy pieces. In: Proc. of ICDT 2007. LNCS, vol. 4353, pp. 1–17. Springer (2007)
6. Gärdenfors, P., Rott, H.: Belief revision. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 4, pp. 35–132. Oxford University Press (1995)
7. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to query answering in $DL-Lite$. In: Proc. of KR 2010. pp. 247–257 (2010)
8. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Proc. of RR 2010 (2010)
9. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Query rewriting for inconsistent $DL-Lite$ ontologies. In: Proc. of RR 2011 (2011)
10. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics X, 133–173 (2008)
11. Savo, D.F., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Romagnoli, V., Ruzzi, M., Stella, G.: MASTRO at work: Experiences on ontology-based data access. In: Proc. of DL 2010. CEUR, ceur-ws.org, vol. 573, pp. 20–31 (2010)

Mixing Open and Closed World Assumption in Ontology-Based Data Access: Non-Uniform Data Complexity

Carsten Lutz¹, İnanç Seylan¹, and Frank Wolter²

¹ Department of Computer Science, University of Bremen, Germany

² Department of Computer Science, University of Liverpool, UK

{clu, seylan}@informatik.uni-bremen.de, wolter@liverpool.ac.uk

Abstract. When using ontologies to access instance data, it can be useful to make a closed world assumption (CWA) for some predicates and an open world assumption (OWA) for others. The main problem with such a setup is that conjunctive query (CQ) answering becomes intractable already for inexpressive description logics such as DL-Lite and \mathcal{EL} . We take a closer look at this situation and carry out a fine-grained complexity analysis by considering the complexity of CQ answering w.r.t. individual TBoxes. Our main results are a dichotomy between AC^0 and $CONP$ for TBoxes formulated in DL-Lite and a dichotomy between P TIME and $CONP$ for \mathcal{EL} -TBoxes. In each tractable case, CQ answering coincides with CQ answering under pure OWA; the CWA might still be useful as it allows queries that are more expressive than CQs.

1 Introduction

Description logics (DLs) increasingly find application in ontology-based data access (OBDA), where an ontology is used to enrich instance data and the chief aim is to provide efficient query answering services. In this context, it is common to make the open world assumption (OWA). Indeed, there are applications where the data is inherently incomplete and the OWA is semantically adequate, for example when the data is extracted from the web. In other applications, however, it is more reasonable to make a closed world assumption (CWA) for some predicates in the data. In particular, when the instance data is taken from a relational database, then the CWA can be appropriate for the data predicates while additional predicates in the ontology should always be interpreted under the OWA (this is the very idea of OBDA). As a concrete example, consider geographical databases such as OpenStreetMap which contain pure geographical data as well as rich annotations, stating for example that a certain polygon describes a ‘popular Thai restaurant’. As argued in [11, 7], it is useful to pursue an OBDA approach to take full advantage of the annotations, where one would naturally interpret the geographical data under the CWA and the annotations under the OWA.

In the DL literature, there are a variety of approaches to adopting the CWA, often based on epistemic operators or rules [6, 8, 10, 19, 22]. In this paper, we adopt the standard semantics from relational databases, which is natural, and straightforward: CWA predicates have to be interpreted exactly as described in the data, assuming standard

(and thus unique) names for data constants; for example, when A is a closed concept name and \mathcal{A} an ABox, then in any model \mathcal{I} of \mathcal{A} we must have $A^{\mathcal{I}} = \{a \mid A(a) \in \mathcal{A}\}$. Note that this semantics is also used in the recently proposed DBoxes [23]. In fact, the setup considered in this paper generalizes both standard OBDA (only OWA predicates permitted) and DBoxes (only CWA predicates permitted in data) by allowing to freely mix OWA and CWA predicates both in the TBox and in the data. For readability, we will from now on speak of open and closed predicates rather than OWA and CWA predicates.

A major problem in admitting closed predicates is that query answering easily becomes intractable regarding data complexity (where the TBox and query are assumed to be fixed and thus of constant size). In fact, this is true already for instance queries (IQs), when only closed predicates are admitted in the data, and for TBoxes formulated in inexpressive DLs such as the core dialect of DL-Lite [5] and \mathcal{EL} [3]; this is shown for conjunctive queries (CQs) and DL-Lite in [9], can easily be transferred to \mathcal{EL} , and strengthened to IQs by adapting a well-known reduction of Schaerf [21]. While this is a relevant and interesting first step, it was recently demonstrated in [15, 16] in the context of standard OBDA with more expressive DLs that a more fine grained, ‘non-uniform’ analysis is possible by studying data complexity on the level of individual TBoxes instead of on the level of logics. In our context, we work with TBoxes of the form (\mathcal{T}, Σ) , where \mathcal{T} is a set of TBox statements as usual and Σ is a set of predicates (concept and role names) that are declared to be closed. We say that CQ answering w.r.t. (\mathcal{T}, Σ) is in PTIME if for every CQ $q(x)$, there exists a polytime algorithm that computes for a given ABox \mathcal{A} the certain answers to q in \mathcal{A} given (\mathcal{T}, Σ) ; CQ answering w.r.t. (\mathcal{T}, Σ) is CONP-hard if there is a Boolean CQ q such that, given an ABox \mathcal{A} , it is CONP-hard to decide whether q is entailed by \mathcal{A} given \mathcal{T} . Other complexities are defined analogously. The main aim of this paper is to carry out a non-uniform analysis of data complexity for query answering with closed predicates in DL-Lite and \mathcal{EL} .

Our main results are a dichotomy between AC^0 and CONP for TBoxes formulated in DL-Lite and a dichotomy between PTIME and CONP for \mathcal{EL} -TBoxes. In each case, we provide a transparent characterization that separates the easy cases from the hard cases. These results are interesting when contrasted with query answering w.r.t. TBoxes that are formulated in the expressive DLs \mathcal{ALC} and \mathcal{ALCC} , where the data complexity is also between AC^0 and CONP, but where the existence of a dichotomy between PTIME and CONP is a deep open question that is equivalent to the Feder-Vardi conjecture for the existence of a dichotomy between PTIME and NP in non-uniform constraint satisfaction problems [16]. We also show that when CQ answering w.r.t. (\mathcal{T}, Σ) is in PTIME, then the certain answers to any CQ q in any ABox \mathcal{A} given (\mathcal{T}, Σ) (which respect the closed-world declarations in Σ) coincide with the open world answers to q in \mathcal{A} given \mathcal{T} —for ABoxes that are satisfiable w.r.t. \mathcal{T} . In a sense, we thus show that CQ answering with closed predicates is *inherently intractable*: in all the tractable and consistent cases, the declaration of closed predicates does not have any impact on query answers.

While this sounds discouraging, there is still a potential benefit of closed predicates in tractable cases: for the ‘closed part’ of the signature, we can go beyond conjunctive queries and admit (almost) full first-order queries without becoming undecidable and,

indeed, without any negative impact on data complexity. We propose a concrete query language that implements this idea and show that AC^0 data complexity is preserved for DL-Lite TBoxes and PTIME data complexity is preserved for \mathcal{EL} -TBoxes when CQs are replaced with queries formulated in the extended language.

Most proofs in this paper are deferred to the (appendix of the) long version, which is available at <http://www.csc.liv.ac.uk/~frank/publ/publ.html>.

2 Preliminaries

We use standard notation from description logic [4]. Let N_C and N_R be countably infinite sets of *concept* and *role names*. A *DL-Lite-concept* is either a concept name from N_C or a concept of the form $\exists r.\top$ or $\exists r^-. \top$, where $r \in N_R$. A *DL-Lite-inclusion* is an expression of the form $B_1 \sqsubseteq B_2$ or $B_1 \sqsubseteq \neg B_2$, where B_1, B_2 are DL-Lite-concepts. A *DL-Lite-TBox* is a finite set of DL-Lite-inclusions. In the literature, this version of DL-Lite is often called $DL-Lite_{core}$. \mathcal{EL} -concepts are constructed according to the rule $C, D := \top \mid A \mid C \sqcap D \mid \exists r.C$, where $A \in N_C$ and $r \in N_R$. An \mathcal{EL} -inclusion is an expression of the form $C \sqsubseteq D$, where C, D are \mathcal{EL} -concepts. An \mathcal{EL} -TBox is a finite set of \mathcal{EL} -inclusions. \mathcal{ELI} extends \mathcal{EL} with the constructor $\exists r^-.C$. ABoxes are finite sets of assertions $A(a)$ and $r(a, b)$ with $A \in N_C$, $r \in N_R$, and a, b individual names. We use $\text{Ind}(\mathcal{A})$ to denote the set of individual names used in the ABox \mathcal{A} and write $r^-(a, b) \in \mathcal{A}$ instead of $r(b, a) \in \mathcal{A}$. We sometimes also use *infinite* ABoxes, but this will be stated explicitly. Interpretations \mathcal{I} are defined as usual, where for the interpretation of individual names we make the standard name assumption (SNA), i.e., $a^{\mathcal{I}} = a$. Note that this implies the unique name assumption (UNA); to avoid enforcing infinite models, we assume that interpretations need not interpret all individual names and use $\text{Ind}(\mathcal{I})$ to denote the individual names interpreted by \mathcal{I} . A concept C (ABox \mathcal{A}) is *satisfiable w.r.t. a TBox \mathcal{T}* if there exists a model \mathcal{I} of \mathcal{T} with $C^{\mathcal{I}} \neq \emptyset$ (that satisfies \mathcal{A} , respectively).

Every ABox \mathcal{A} corresponds to an interpretation $\mathcal{I}_{\mathcal{A}}$ whose domain is $\text{Ind}(\mathcal{A})$ and in which $a \in A^{\mathcal{I}_{\mathcal{A}}}$ iff $A(a) \in \mathcal{A}$, for all $A \in N_C$ and $a \in \text{Ind}(\mathcal{A})$ and similarly for role names. Conversely, every interpretation \mathcal{I} corresponds to a (possibly infinite) ABox $\mathcal{A}_{\mathcal{I}}$ whose individual names are $\Delta^{\mathcal{I}}$.

A *predicate* is a concept or role name. A *signature* Σ is a finite set of predicates. The signature $\text{sig}(C)$ of a concept C , $\text{sig}(\mathcal{T})$ of a TBox \mathcal{T} , and $\text{sig}(\mathcal{A})$ of an ABox \mathcal{A} , is the set of predicates occurring in C , \mathcal{T} , and \mathcal{A} , respectively. For being able to declare predicates as closed, we add an additional component to TBoxes. A pair (\mathcal{T}, Σ) with \mathcal{T} a TBox \mathcal{T} and Σ a signature is a *TBox with closed predicates*. For any ABox \mathcal{A} , a *model \mathcal{I} of (\mathcal{T}, Σ)* and \mathcal{A} is an interpretation \mathcal{I} with $\text{Ind}(\mathcal{A}) \subseteq \text{Ind}(\mathcal{I})$ that satisfies \mathcal{T} and \mathcal{A} and such that

$$\begin{aligned} A^{\mathcal{I}} &= \{a \mid A(a) \in \mathcal{A}\} && \text{for all } A \in \Sigma \cap N_C \\ r^{\mathcal{I}} &= \{(a, b) \mid r(a, b) \in \mathcal{A}\} && \text{for all } r \in \Sigma \cap N_R. \end{aligned}$$

Note that TBox statements that only involve closed predicates are effectively integrity constraints in the standard database sense [1]. In a DL context, integrity constraints are discussed for example in [6, 8, 17–19].

A *first-order query (FOQ)* $q(\mathbf{x})$ is a first-order formula constructed from atoms $A(t)$, $r(t, t')$, and $t = t'$, where t, t' range over individual names and variables and $\mathbf{x} = x_1, \dots, x_k$ contains all free variables of q . We call \mathbf{x} the *answer variables of $q(\mathbf{x})$* . A *conjunctive query (CQ)* $q(\mathbf{x})$ is a FOQ using conjunction and existential quantification, only. A tuple $\mathbf{a} = a_1, \dots, a_k \subseteq \text{Ind}(\mathcal{A})$ is a *certain answer to $q(\mathbf{x})$ in \mathcal{A} given (\mathcal{T}, Σ)* , in symbols $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} q(\mathbf{a})$, if $\mathcal{I} \models q[a_1, \dots, a_k]$ for all models \mathcal{I} of (\mathcal{T}, Σ) and \mathcal{A} . When computing certain answers we assume that all individual names in the query occur in the ABox. If $\Sigma = \emptyset$, then we simply omit Σ and write $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$ instead of $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} q(\mathbf{a})$. If C is an \mathcal{ELI} -concept, then the CQ corresponding to $C(a)$ is defined in the usual way and called an \mathcal{ELI} -instance query. \mathcal{EL} -instance queries are defined analogously.

The following definition generalizes the definition of non-uniform data complexity introduced in [16] to TBoxes with closed predicates.

Definition 1. *Let (\mathcal{T}, Σ) be a TBox with closed predicates. Then*

- CQ answering w.r.t. (\mathcal{T}, Σ) is in PTIME if for every CQ $q(\mathbf{x})$ there is a polytime algorithm that computes, for a given ABox \mathcal{A} , all $\mathbf{a} \subseteq \text{Ind}(\mathcal{A})$ with $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} q(\mathbf{a})$;
- CQ answering w.r.t. (\mathcal{T}, Σ) is coNP-hard if there is a Boolean CQ q such that it is coNP-hard to decide, given an ABox \mathcal{A} , whether $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} q$.

For other classes of queries such as FOQs and \mathcal{ELI} -instance queries, analogous notions can be defined. It is known that for $\Sigma = \emptyset$, CQ answering is in PTIME for \mathcal{EL} -TBoxes [5, 13] and in AC^0 for DL-Lite [5, 2]. FOQ-answering is undecidable even for the empty TBox, due to the OWA.

The following property, plays a central role in our analysis; see [16] which also makes intensive use of this notion (there called ABox disjunction property).

Definition 2 (Disjunction property). *A TBox with closed predicates (\mathcal{T}, Σ) has the disjunction property if for all ABoxes \mathcal{A} and \mathcal{ELI} -instance queries $C_1(a)$ and $C_2(a)$, $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} C_1(a) \vee C_2(a)$ implies $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} C_i(a)$ for some $i \in \{1, 2\}$.*

It is standard to show that DL-Lite and \mathcal{EL} TBoxes *without* closed predicates have the disjunction property [16].

3 Main Results and Illustrating Examples

We first formulate the dichotomy result for DL-Lite. The next definition introduces a class of TBoxes with closed predicates that turn out to be exactly the TBoxes for which query answering is in AC^0 .

Definition 3. *A DL-Lite TBox \mathcal{T} with closed predicates Σ is safe if there are no DL-Lite-concepts B_1, B_2 and role r such that*

1. B_1 is satisfiable w.r.t. \mathcal{T} ;
2. $\mathcal{T} \models B_1 \sqsubseteq \exists r. \top$ and $\mathcal{T} \models \exists r^-. \top \sqsubseteq B_2$; and

3. $B_1 \neq \exists r. \top$, $\text{sig}(B_2) \subseteq \Sigma$, and $\text{sig}(r) \cap \Sigma = \emptyset$.³

Note that it is easy to check in PTIME whether a given DL-Lite TBox is safe since subsumption in DL-Lite can be decided in PTime [5]. Our results concerning DL-Lite are summarized by the following theorem.

Theorem 1 (DL-Lite dichotomy). *Let (\mathcal{T}, Σ) be a DL-Lite-TBox with closed predicates. Then the following holds:*

1. *If (\mathcal{T}, Σ) is not safe, then the disjunction property fails and there is an \mathcal{EL} -instance query $C(a)$ such that answering $C(a)$ w.r.t. (\mathcal{T}, Σ) is coNP-hard.*
2. *If (\mathcal{T}, Σ) is safe, then*
 - (a) *CQ answering w.r.t. (\mathcal{T}, Σ) coincides with CQ answering w.r.t. (\mathcal{T}, \emptyset) for all ABoxes that are satisfiable w.r.t. (\mathcal{T}, Σ) , i.e., for every CQ $q(\mathbf{x})$ and $\mathbf{a} \subseteq \text{Ind}(\mathcal{A})$, we have $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} q(\mathbf{a})$ iff $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$.*
 - (b) *CQ answering w.r.t. (\mathcal{T}, Σ) is in AC^0 and (\mathcal{T}, Σ) has the disjunction property.*

The following example illustrates Theorem 1.

Example 1. (a) Let $\mathcal{T} = \{A \sqsubseteq \exists r. \top, \exists r^- . \top \sqsubseteq B\}$ and $\Sigma = \{B\}$. (\mathcal{T}, Σ) is not safe. The disjunction property can be refuted as follows. Let $\mathcal{A} = \{A(a), B(b_1), A_1(b_1), B(b_2), A_2(b_2)\}$, where A_1, A_2 are fresh concept names. Then

1. $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} \exists r. (A_1 \sqcap B)(a) \vee \exists r. (A_2 \sqcap B)(a)$;
2. $\mathcal{T}, \mathcal{A} \not\models_{c(\Sigma)} \exists r. (A_i \sqcap B)(a)$ for $i = 1, 2$.

Point 1 should be clear since in any model \mathcal{I} of (\mathcal{T}, Σ) and \mathcal{A} one has to link a with r to b_1 or to b_2 to satisfy \mathcal{T} . For Point 2 and $i \in \{1, 2\}$, consider the model \mathcal{I}_i that corresponds to \mathcal{A} expanded with $r(a, b_i)$. Then \mathcal{I}_i is a model of (\mathcal{T}, Σ) and \mathcal{A} (note that $r \notin \Sigma$) but $a \notin (\exists r. (A_{\bar{i}} \sqcap B))^{\mathcal{I}_i}$ where $\bar{1} = 2$ and $\bar{2} = 1$. Thus $\mathcal{T}, \mathcal{A} \not\models_{c(\Sigma)} \exists r. (A_i \sqcap B)(a)$. When we add any of A, A_1, A_2 to Σ , all statements are still true.

(b) The failure of the disjunction property for the ABox \mathcal{A} and TBox \mathcal{T} results in a choice that enables a coNP-hardness proof by reduction of 2+2-SAT, a variant of propositional satisfiability where each clause contains precisely two positive literals and two negative literals [21]. For this reduction, it suffices to use an \mathcal{EL} -query that uses the above queries $\exists r. (A_i \sqcap B)(a)$, $i = 1, 2$, as subqueries for encoding truth values.

The proof of Theorem 1 is given in the next section. We now come to the case of TBoxes formulated in \mathcal{EL} , where we start with examples. Observe that we can find an \mathcal{EL} -TBox without the disjunction property by a reformulation of the DL-Lite TBox from Example 1. Let $\mathcal{T}' = \{A \sqsubseteq \exists r. B\}$ with $\Sigma = \{B\}$. Then, in the same way as for (\mathcal{T}, Σ) one can show that the disjunction property fails for (\mathcal{T}', Σ) and that CQ answering is coNP-hard. In \mathcal{EL} , however, there is an additional (and more subtle) cause for non-tractability, which we discuss in the following example.

Example 2. Consider again $\mathcal{T}' = \{A \sqsubseteq \exists r. B\}$, but now set $\Sigma' = \{r\}$. We first show that (\mathcal{T}', Σ') does not have the disjunction property. Let $\mathcal{A}' = \{A(a), r(a, b_1), A_1(b_1), r(a, b_2), A_2(b_2)\}$, where A_1, A_2 are fresh concept names. Then one can easily show that the disjunction property fails:

³ In other words, r is a role name and $r \notin \Sigma$ or $r = s^-$ for a role name $s \notin \Sigma$.

- $\mathcal{T}', \mathcal{A}' \models_{c(\Sigma')} \exists r.(A_1 \sqcap B)(a) \vee \exists r.(A_2 \sqcap B)(a)$;
- $\mathcal{T}', \mathcal{A}' \not\models_{c(\Sigma')} \exists r.(A_i \sqcap B)(a)$, for $i = 1, 2$.

It is crucial that $B \notin \Sigma'$. The proof of CONP-hardness is very similar to the proof mentioned in Example 1.

Observe that one cannot reproduce this example in DL-Lite: for the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.\top, \exists r.\top \sqsubseteq B\}$ with $\Sigma' = \{r\}$, we have $\mathcal{T}, \mathcal{A}' \models_{c(\Sigma')} B(b_i)$ for $i = 1, 2$ and, therefore, $\mathcal{T}, \mathcal{A}' \models_{c(\Sigma')} \exists r.(A_i \sqcap B)(a)$, for $i = 1, 2$. Thus, the disjunction property is not violated.

We now identify a class of \mathcal{EL} -TBoxes with closed predicates that turn out to be exactly the TBoxes for which CQ answering is in PTIME. We call a concept E a *top-level conjunct (tlc)* of C if C is of the form $D_1 \sqcap \dots \sqcap D_n$ with $n \geq 1$ and $E = D_i$ for some i .

Definition 4. Let (\mathcal{T}, Σ) be an \mathcal{EL} -TBox with closed predicates. (\mathcal{T}, Σ) is safe if there exists no \mathcal{EL} -inclusion $C \sqsubseteq \exists r.D$ such that

1. $\mathcal{T} \models C \sqsubseteq \exists r.D$;
2. there does not exist a tlc $\exists r.C'$ of C with $\mathcal{T} \models C' \sqsubseteq D$;
3. one of the following is true:
 - (s1) $r \notin \Sigma$ and $\text{sig}(D) \cap \Sigma \neq \emptyset$;
 - (s2) $r \in \Sigma$, $\text{sig}(D) \not\subseteq \Sigma$ and there is no Σ -concept E with $\mathcal{T} \models C \sqsubseteq \exists r.E$ and $\mathcal{T} \models E \sqsubseteq D$.

Note that Condition 3(s1) of Definition 4 is similar to the definition of safety for DL-Lite. Example 2 shows why Condition 3(s2) is needed. The following example illustrates the additional requirement of 3(s2) that no “interpolating” Σ -concept E exists.

Example 3. Let $\mathcal{T} = \{A \sqsubseteq \exists r.E, E \sqsubseteq B\}$ and first assume that $\Sigma = \{r\}$. Then the inclusion $A \sqsubseteq \exists r.B$ satisfies Condition 3(s2) and thus (\mathcal{T}, Σ) is not safe. Now assume $\Sigma = \{r, E\}$. Then, the inclusion $A \sqsubseteq \exists r.B$ does not violate safety because E can be used as a ‘ Σ -interpolant’. Note that the ABox \mathcal{A}' from Example 2, which we used to refute the disjunction property in a very similar situation, is simply unsatisfiable w.r.t. (\mathcal{T}, Σ) because E has to be interpreted as the empty set. Indeed, it can be shown that (\mathcal{T}, Σ) is safe.

Note that, unlike the DL-Lite case, the definition of safety of \mathcal{EL} -TBoxes with closed predicates does not immediately suggest a decision procedure since there are infinitely many candidates for the concepts C , D , and E . We conjecture that safety is decidable in PTIME, but pursuing this further is left for future work.

Theorem 2 (\mathcal{EL} -dichotomy). Let (\mathcal{T}, Σ) be an \mathcal{EL} -TBox with closed predicates. Then the following holds:

1. If (\mathcal{T}, Σ) is not safe, then the disjunction property fails and there exists an \mathcal{EL} -instance query $C(a)$ such that answering $C(a)$ w.r.t. (\mathcal{T}, Σ) is coNP-hard.
2. If (\mathcal{T}, Σ) is safe, then
 - (a) CQ answering w.r.t. (\mathcal{T}, Σ) coincides with CQ answering w.r.t. (\mathcal{T}, \emptyset) for all ABoxes that are satisfiable w.r.t. (\mathcal{T}, Σ) .

- (b) *CQ answering w.r.t. (\mathcal{T}, Σ) is in PTIME and (\mathcal{T}, Σ) has the disjunction property.*

As noted in the introduction, Theorems 1 and 2 essentially show that CQ answering with closed predicates is inherently intractable. Note, though, that Points 2(a) of these theorems refer only to satisfiable ABoxes. In fact, TBox statements that refer only to closed predicates act as integrity constraints also in the tractable cases. Moreover, safe TBoxes admit *any* integrity constraint that can be formulated in the DL at hand, i.e., if a TBox \mathcal{T} formulated in DL-Lite or \mathcal{EL} is safe, then it is still safe after adding any concept inclusions that refers only to closed predicates. In the appendix of the long version, we show that checking satisfiability of ABoxes w.r.t. safe TBoxes is in AC^0 for DL-Lite and in PTIME for \mathcal{EL} (for data complexity).

Another way of taking advantage of closed predicates without losing tractability is to admit more expressive query languages. Indeed, mixing open and closed predicates seems particularly useful when large parts of the data stem from a relational database, as in the geographical database application mentioned in the introduction. In such a setup, one would typically not want to give up FOQs (SQL queries) available in the relational system to accomodate open world predicates. We propose a query language that combines, in a straightforward way, FOQs for closed predicates with CQs for open (and closed) predicates. We then show that, for safe TBoxes with closed predicates, such queries can be answered as efficiently as CQs both in the case of DL-Lite and of \mathcal{EL} .

As in the relational database setting, we allow only FOQs that are domain-independent and thus correspond to expressions of relational algebra (and SQL queries). Formally, a FOQ $q(\mathbf{x})$ is *domain-independent* if for all interpretations \mathcal{I} and \mathcal{J} such that $P^{\mathcal{I}} = P^{\mathcal{J}}$ for all $P \in \text{sig}(q(\mathbf{x}))$, we have $\mathcal{I} \models q[\mathbf{d}]$ iff $\mathcal{J} \models q[\mathbf{d}]$ for all tuples $\mathbf{d} \subseteq \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}$. Intuitively, the truth value of a domain-independent FOQ depends only on the interpretation of the data predicates, but not on the actual domain of the interpretation. For example, $\neg A(x)$, is not domain-independent whereas $B(x) \wedge \neg A(x)$ is domain-independent. In our query language, we allow domain-independent FOQs over closed predicates as atoms in CQs.

Definition 5. *Let Σ be a signature that declares closed predicates. A conjunctive query with $FO(\Sigma)$ plugins (abbreviated $CQ^{FO(\Sigma)}$) is of the form $\exists x_1 \dots \exists x_n (\varphi_1 \wedge \dots \wedge \varphi_m)$, where $n \geq 0$, $m \geq 1$, and each φ_i is either a CQ or a domain-independent FOQ whose signature is included in Σ .*

The subsequent theorem shows that switching from CQs to $CQ^{FO(\Sigma)}$ s does not increase data complexity.

Theorem 3.

1. *$CQ^{FO(\Sigma)}$ -answering w.r.t. safe DL-Lite-TBoxes with closed predicates is in AC^0 . More precisely, for every such TBox (\mathcal{T}, Σ) and $CQ^{FO(\Sigma)}$ $q(\mathbf{x})$, there exists a FOQ $q'(\mathbf{x})$ such that for all \mathcal{A} and $\mathbf{a} \subseteq \text{Ind}(\mathcal{A})$: $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} q(\mathbf{a})$ iff $\mathcal{I}_{\mathcal{A}} \models q'(\mathbf{a})$.*
2. *$CQ^{FO(\Sigma)}$ -answering w.r.t. safe \mathcal{EL} -TBoxes is in PTIME.*

Query languages between CQs and FOQs have been studied before. In the standard setup where all predicates are open, it was shown in [20] that extending CQs with union and atomic negation results in coNP-hardness both in DL-Lite and in \mathcal{EL} , and that extending CQs with union and inequality results in undecidability in \mathcal{EL} . In our query language $\text{CQ}^{\text{FO}(\Sigma)}$, we avoid these problems by allowing only CQs for the open predicates while restricting the expressive power of FOQs (which admits disjunction, full negation, and (in)equality) to closed predicates. The language EQL-Lite(CQ) proposed in [6] can, in some sense, be viewed as a fragment of our language that admits the full expressivity of FOQs, but in which only closed predicates are admitted. Note though, that the EQL-Lite approach closes predicates only for querying while all predicates are interpreted as open world for TBox reasoning.

4 Proof Sketches

We sketch proofs of Theorems 1 and 2. We begin with the first part of Theorem 1.

Lemma 1. *If a DL-Lite-TBox \mathcal{T} with closed predicates Σ is not safe, then the disjunction property fails and there exists an \mathcal{ELI} -instance query $C(a)$ such that answering $C(a)$ w.r.t. (\mathcal{T}, Σ) is coNP-hard.*

Proof. Assume that $B_1 \sqsubseteq \exists r.\top, \exists r^-. \top \sqsubseteq B_2$ satisfy the conditions of Definition 3. Take a finite model \mathcal{I} of \mathcal{T} with $a_0 \in B_1^{\mathcal{I}}$; such a model exists since B_1 is satisfiable w.r.t. \mathcal{T} and DL-Lite has the finite model property. Let $S = \{b \in \Delta^{\mathcal{I}} \mid (a_0, b) \in r^{\mathcal{I}}\}$. Since $B_1 \neq \exists r.\top$, we have $a_0 \in B_1^{\mathcal{I}_S}$ for the interpretation \mathcal{I}_S obtained from \mathcal{I} by removing all pairs (a_0, b) with $b \in S$ from $r^{\mathcal{I}}$. Take the ABox \mathcal{A}_S corresponding to \mathcal{I}_S and let \mathcal{A} be the disjoint union of two copies of \mathcal{A}_S . We denote the individual names of the first copy by $(b, 1)$, $b \in \Delta^{\mathcal{I}}$, and the elements of the second copy by $(b, 2)$, $b \in \Delta^{\mathcal{I}}$. Let

$$\mathcal{A}' = \mathcal{A} \cup \{A_1(b, 1) \mid b \in B_2^{\mathcal{I}}\} \cup \{A_2(b, 2) \mid b \in B_2^{\mathcal{I}}\},$$

where A_1 and A_2 are fresh concept names. Now one can show that the disjunction property fails: $(\mathcal{T}, \mathcal{A}') \models_{c(\Sigma)} \exists r.(A_1 \sqcap B_2)(a_0, 1) \vee \exists r.(A_2 \sqcap B_2)(a_0, 1)$ and $(\mathcal{T}, \mathcal{A}') \not\models_{c(\Sigma)} \exists r.(A_i \sqcap B_2)(a_0, 1)$ for $i = 1, 2$.

The CONP-hardness proof is now similar to the proof for Example 1 given in the appendix of the long version. \square

For the second part of Theorem 1, we first prove (a):

Lemma 2. *Let (\mathcal{T}, Σ) be a DL-Lite TBox with closed predicates. If (\mathcal{T}, Σ) is safe, then CQ answering w.r.t. (\mathcal{T}, Σ) coincides with CQ answering w.r.t. \mathcal{T} without closed predicates for ABoxes that are satisfiable w.r.t. (\mathcal{T}, Σ) .*

Proof. Let (\mathcal{T}, Σ) be safe and assume that \mathcal{A} is satisfiable w.r.t. (\mathcal{T}, Σ) . We remind the reader of the construction of a canonical model \mathcal{I} of \mathcal{T} and \mathcal{A} (without closed predicates!) [12]. \mathcal{I} is the interpretation corresponding to an ABox \mathcal{A}_c that is the limit of a sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \dots$. Let $\mathcal{A}_0 = \mathcal{A}$ and assume a_0, \dots is an infinite list of individual names such that $\text{Ind}(\mathcal{A}_0) = \{a_0, \dots, a_k\}$. Assume \mathcal{A}_j has been defined already. Let i be minimal such that there exists $B_1 \sqsubseteq B_2 \in \mathcal{T}$ with $\mathcal{A}_j \models B_1(a_i)$ but $\mathcal{A}_j \not\models B_2(a_i)$ (if no such i exists, then set $\mathcal{A}_c := \mathcal{A}_j$). Then

- if B_2 is a concept name, let $\mathcal{A}_{j+1} = \mathcal{A}_j \cup \{B_2(a_i)\}$;
- if $B_2 = \exists s.\top$, then take a fresh individual $b_{a_i,s}$ and set $\mathcal{A}_{j+1} = \mathcal{A}_j \cup \{s(a_i, b_{a_i,s})\}$.

Now let \mathcal{J} be the interpretation corresponding to the ABox $\mathcal{A}_c = \bigcup_{i \geq 0} \mathcal{A}_i$. It is known that \mathcal{J} is a model of $(\mathcal{T}, \mathcal{A})$ with the following properties:

1. For all CQs $q(x)$ and $\mathbf{a} \subseteq \text{Ind}(\mathcal{A})$: $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$ iff $\mathcal{J} \models q[\mathbf{a}]$.
2. For any individual $b_{a_i,s} \in \text{Ind}(\mathcal{A}_c) \setminus \text{Ind}(\mathcal{A})$ introduced as a witness for some $B_2 = \exists s.\top$, we have $B(b_{a_i,s}) \in \mathcal{A}_c$ iff $\mathcal{T} \models \exists s^-. \top \sqsubseteq B$, for every DL-Lite-concept B .

To show that \mathcal{J} is a model of (\mathcal{T}, Σ) and \mathcal{A} we prove the following

Claim 1. For all $i \geq 0$ and for all $a \in \text{Ind}(\mathcal{A}_i)$, if $B_1 \sqsubseteq B_2 \in \mathcal{T}$ with $\mathcal{A}_i \models B_1(a)$ but $\mathcal{A}_i \not\models B_2(a)$, then $\text{sig}(B_2) \cap \Sigma = \emptyset$.

Claim 1 holds for all \mathcal{A}_i , $i \geq 0$, and all $a \in \text{Ind}(\mathcal{A})$: otherwise, \mathcal{A}_{i+1} is unsatisfiable, in contradiction to Point 1. It follows that Claim 1 holds for $i = 0$. We proceed by induction, assuming that Claim 1 has been proved for \mathcal{A}_i , but that to the contrary of what is to be shown there are $B_1 \sqsubseteq B_2 \in \mathcal{T}$ with $\mathcal{A}_{i+1} \models B_1(a)$, $\mathcal{A}_{i+1} \not\models B_2(a)$, and $\text{sig}(B_2) \cap \Sigma \neq \emptyset$, i.e., $\text{sig}(B_2) \subseteq \Sigma$. By what was said above, we have $a \notin \text{Ind}(\mathcal{A})$ and thus a was introduced as a witness for some $\exists s.\top$. By IH, $\text{sig}(s) \cap \Sigma = \emptyset$ and there exists $B_1 \neq \exists s.\top$ with $B_1 \sqsubseteq \exists s.\top \in \mathcal{T}$. Point 2 yields $\mathcal{T} \models \exists s^-. \top \sqsubseteq B_2$, in contrary to (\mathcal{T}, Σ) being safe. This finishes the proof of Claim 1.

It follows from Claim 1 that \mathcal{J} is a model of (\mathcal{T}, Σ) and \mathcal{A} . Thus, we have for CQs $q(x)$ and $\mathbf{a} \subseteq \text{Ind}(\mathcal{A})$: if $\mathcal{T}, \mathcal{A} \not\models q(\mathbf{a})$, then $\mathcal{J} \not\models q[\mathbf{a}]$, and so $\mathcal{T}, \mathcal{A} \not\models_{c(\Sigma)} q(\mathbf{a})$, as required. \square

To obtain a proof of Part 2 of Theorem 1, it remains to show that, for safe (\mathcal{T}, Σ) , it is in AC^0 to decide whether an ABox is satisfiable w.r.t. (\mathcal{T}, Σ) . To this end, it is readily checked that an ABox \mathcal{A} which is satisfiable w.r.t. \mathcal{T} is satisfiable w.r.t. a safe (\mathcal{T}, Σ) iff $\mathcal{T}, \mathcal{A} \models B(a)$ implies $\mathcal{A} \models B(a)$ for all DL-Lite concepts B over Σ . To see that this condition is in AC^0 , let $\varphi_B(x)$ be an FO query with $\mathcal{T}, \mathcal{A} \models B(a)$ iff $\mathcal{I}_{\mathcal{A}} \models \varphi_B(a)$, where $\mathcal{I}_{\mathcal{A}}$ is the interpretation corresponding to \mathcal{A} . Then \mathcal{A} is not satisfiable w.r.t. (\mathcal{T}, Σ) iff $\mathcal{I}_{\mathcal{A}} \models \exists x \bigvee_{B \in X} (\varphi_B(x) \wedge \neg B(x))$ where X denotes the set of all DL-Lite concepts over Σ .

We now come to Theorem 2. With the exception of proof steps involving Condition 3(s2) of Definition 4, the proof technique for Theorem 2 extends the proof technique introduced for DL-Lite. We therefore focus on 3(s2). We require a certain interpolation property. This interpolation property has been studied before for \mathcal{ALC} and several of its extensions in the context of query rewriting for DBoxes and Beth definability [23, 24]. Note that it is different from the interpolation property investigated in [14], which requires the interpolant to be a TBox instead of a concept.

Lemma 3 (Interpolation). *Let $\mathcal{T}_1, \mathcal{T}_2$ be \mathcal{EL} -TBoxes, $\mathcal{T}_1 \cup \mathcal{T}_2 \models D_0 \sqsubseteq D_1$ with $\text{sig}(\mathcal{T}_1, D_0) \cap \text{sig}(\mathcal{T}_2, D_1) \subseteq \Sigma$. Then there exists a Σ -concept F such that $\mathcal{T}_1 \cup \mathcal{T}_2 \models D_0 \sqsubseteq F$ and $\mathcal{T}_1 \cup \mathcal{T}_2 \models F \sqsubseteq D_1$.*

The following lemma is the crucial step for proving Part 1 of Theorem 2 if 3(s2) applies.

Lemma 4. *Let (\mathcal{T}, Σ) be an \mathcal{EL} -TBox with closed predicates such that safety is violated by $C \sqsubseteq \exists r.D \in \mathcal{T}$ because $\exists(s2)$ holds. Then the disjunction property fails.*

Proof. Assume $C \sqsubseteq \exists r.D$ is given. Take the canonical model $\mathcal{I}_{\mathcal{T}, C}$ of \mathcal{T} and C as defined in [14] (its domain $\Delta^{\mathcal{I}_{\mathcal{T}, C}}$ consists of names a_F , F a subconcept of \mathcal{T} or C , and $a_F \in G^{\mathcal{I}_{\mathcal{T}, C}}$ iff $\mathcal{T} \models F \sqsubseteq G$, for all \mathcal{EL} -concepts G). Assume for simplicity that $(a_F, a_C) \notin r^{\mathcal{I}_{\mathcal{T}, C}}$ for any $a_F \in \Delta^{\mathcal{I}_{\mathcal{T}, C}}$. Let

$$S = \{a_G \in \Delta^{\mathcal{I}_{\mathcal{T}, C}} \mid (a_C, a_G) \in r^{\mathcal{I}_{\mathcal{T}, C}}, \exists r.G \text{ is not a top level conjunct of } C\}$$

and let \mathcal{I}_S be the interpretation obtained from $\mathcal{I}_{\mathcal{T}, C}$ by removing all pairs (d, d') with $d' \in S$ from $r^{\mathcal{I}_{\mathcal{T}, C}}$. We have $a_C \in C^{\mathcal{I}_S}$. Let \mathcal{A}_S be the ABox corresponding to \mathcal{I}_S and

$$K = \{G \mid \exists r.G \in \text{sub}(\mathcal{T}), \mathcal{T} \models C \sqsubseteq \exists r.G\}.$$

Since there is no tlc C' of C with $\mathcal{T} \models C' \sqsubseteq D$, by a result of [14] (Lemma 16), there exists $G \in K$ with $\mathcal{T} \models G \sqsubseteq D$.

Introduce copies X^0 and X^1 of any non- Σ -predicate X . Denote by E^0 and E^1 the resulting concept if each non- Σ predicate X in E is replaced by X^0 and, respectively, X^1 . Similarly, denote by \mathcal{T}^0 and \mathcal{T}^1 the TBoxes obtained from \mathcal{T} by replacing all concepts E in \mathcal{T} by E^0 and E^1 , respectively. The following can be proved using Lemma 3:

Fact. For all $G \in K$: $\mathcal{T}^0 \cup \mathcal{T}^1 \not\models G^0 \sqsubseteq D^1$.

Now one can take the canonical models $\mathcal{J}_G := \mathcal{I}_{\mathcal{T}^0 \cup \mathcal{T}^1, G^0}$ for any $G \in K$ and obtain for $a_G := a_{G^0}$ that $a_G \notin (D^1)^{\mathcal{J}_G}$. Let $\mathcal{A}_{G, \Sigma}$ be the Σ -reduct of the ABox corresponding to \mathcal{J}_G and assume that the $\text{Ind}(\mathcal{A}_{G, \Sigma})$ are mutually disjoint, for $G \in K$, and that $a_G \in \text{Ind}(\mathcal{A}_{G, \Sigma})$, for all $G \in K$. Introduce two copies $\mathcal{A}_{G, \Sigma}^1$ and $\mathcal{A}_{G, \Sigma}^2$ of $\mathcal{A}_{G, \Sigma}$, for $G \in K$. We denote the elements of the first copy by $(a, 1)$, for $a \in \text{Ind}(\mathcal{A}_{G, \Sigma})$ and the elements of the second copy by $(a, 2)$, for $a \in \text{Ind}(\mathcal{A}_{G, \Sigma})$. Define the ABox \mathcal{A} by taking two fresh concept names A_1 and A_2 and the union of $\mathcal{A}_S \cup \bigcup_{G \in K} \mathcal{A}_{G, \Sigma}^1 \cup \mathcal{A}_{G, \Sigma}^2$ and the assertions $r(a_C, (a_G, 1)), r(a_C, (a_G, 2)), A_1(a_G, 1)$, and $A_2(a_G, 2)$, for every $G \in K$ and $A_1(a_{D'})$, for every tlc $\exists r.D'$ of C . One can show that the disjunction property is violated: $\mathcal{T}, \mathcal{A} \models_{c(\Sigma)} \exists r.(A_1 \sqcap D)(a_C) \vee \exists r.(A_2 \sqcap D)(a_C)$ and $\mathcal{T}, \mathcal{A} \not\models_{c(\Sigma)} \exists r.(A_i \sqcap D)(a_C)$, for $i = 1, 2$. \square

5 Future Work

We have presented first results regarding the non-uniform complexity of query answering in the presence of open and closed world predicates. We expect the proposed extension of CQs with FO-plugins to be useful in practical applications where closed predicates are important and safe TBoxes suffice. As future work, we plan to extend our investigation to more expressive DLs. For example, we conjecture that transparent dichotomy results can still be obtained for the extensions of DL-Lite_{core} and \mathcal{EL} with role hierarchies.

Acknowledgments. Carsten Lutz and İnanç Seylan were supported by the DFG SFB/TR 8 “Spatial Cognition”.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyashev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research*. (JAIR), 36:1–69, 2009.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the \mathcal{EL} envelope. In *IJCAI*, pages 364–369, 2005.
4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge Univ. Press, 2003.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.
6. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. EQL-Lite: Effective first-order query processing in description logics. In *IJCAI*, pages 274–279, 2007.
7. M. Codrescu, G. Horsinka, O. Kutz, T. Mossakowski, and R. Rau. DO-ROAM: Activity-oriented search and navigation with OpenStreetMap. In *GeoSpatial Semantics*, volume 6631 of LNCS, pages 88–107. Springer, 2011.
8. F. M. Donini, D. Nardi, and R. Rosati. Description logics of minimal knowledge and negation as failure. *ACM Transactions on Computational Logic*, 3(2):177–225, 2002.
9. E. Franconi, Y. A. Ibáñez-García, and I. Seylan. Query answering with DBoxes is hard. *Electronic Notes on Theoretical Computer Science*, 278:71–84, 2011.
10. S. Grimm and B. Motik. Closed world reasoning in the semantic web through epistemic operators. In *OWLED*, volume 188 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.
11. S. Hübner, R. Spittel, U. Visser, and T. J. Vögele. Ontology-based search for interactive digital maps. *IEEE Intelligent Systems*, 19(3):80–86, 2004.
12. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyashev. The combined approach to query answering in DL-Lite. In *KR*, 2010.
13. C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic \mathcal{EL} using a relational database system. In *IJCAI*, pages 2070–2075, 2009.
14. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic \mathcal{EL} . *Journal of Symbolic Computation*, 45(2):194–228, 2010.
15. C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In *Description Logics*, 2011.
16. C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In *KR*, 2012.
17. A. Mehdi, S. Rudolph, and S. Grimm. Epistemic querying of OWL knowledge bases. In *ESWC*, volume 6643 of LNCS, pages 397–409. Springer, 2011.
18. B. Motik, I. Horrocks, and U. Sattler. Bridging the gap between OWL and relational databases. *Journal of Web Semantics*, 7(2):74–89, 2009.
19. B. Motik and R. Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.
20. R. Rosati. The limits of querying ontologies. In *ICDT*, pages 164–178, 2007.
21. A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.
22. K. Sengupta, A. A. Krisnadhi, and P. Hitzler. Local closed world semantics: Grounded circumscription for OWL. In *ISWC*, volume 7031 of LNCS, pages 617–632. Springer, 2011.
23. I. Seylan, E. Franconi, and J. de Bruijn. Effective query rewriting with ontologies over DBoxes. In *IJCAI*, pages 923–929, 2009.
24. B. ten Cate, E. Franconi, and I. Seylan. Beth definability in expressive description logics. In *IJCAI*, pages 1099–1106, 2011.

Modelling Structured Domains Using Description Graphs and Logic Programming*

Despoina Magka, Boris Motik, and Ian Horrocks

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, OX1 3QD, UK

1 Introduction

OWL 2¹ is commonly used to represent objects with complex structure, such as complex assemblies in engineering applications [5], human anatomy [13], or the structure of chemical molecules [7]. In order to ground our discussion, we next present a concrete application of the latter kind; however, the problems and the solution that we identify apply to numerous similar scenarios.

The European Bioinformatics Institute (EBI) has developed the ChEBI² ontology—a public dictionary of *molecular entities* used to enhance interoperability of applications supporting tasks such as drug discovery. In order to automate the classification of molecular entities, ChEBI descriptions were translated into OWL and then classified using automated reasoning. However, this approach was hindered by the fundamental inability of OWL to precisely represent the structure of complex molecular entities, as the tree-model property of OWL prevents one from describing non-tree-like relationships using schema axioms. For example, OWL axioms can state that butane molecules have four carbon atoms, but they cannot state that the four atoms in a cyclobutane molecule are arranged in a ring. Please note that this applies to *schema* descriptions only: the structure of a particular cyclobutane molecule can be represented using class and property assertions, but the general definition of *all* cyclobutane molecules—a problem that terminologies such as ChEBI aim to solve—cannot be fully described in OWL. As we show in Section 3, an ontology may therefore fail to entail certain desired consequences.

A common solution to this problem is to extend OWL 2 with a rule-based formalism such as SWRL;³ however, this either results in undecidability [9] or requires restrictions in the shape of the rules [8], which typically prevent the rules from axiomatising the required structures. An alternative approach suggests a combination of OWL 2, rules, and *description graphs* (DGs) [12]—a graphical notation for describing non-tree-like structures. Decidability of reasoning is ensured via a *property separation* condition and by requiring DGs to be *acyclic*. Intuitively, the latter means that DGs can describe structures of arbitrary shape, but bounded in size, while the former limits the interaction between the OWL and DG parts, thus preventing multiple DG structures from merging into one structure of (potentially) unbounded size. As reported in [7], DGs solved

* Work supported by EU FP7 SEALS and EPSRC projects ConDOR, ExODA, and LogMap.

¹ <http://www.w3.org/TR/owl2-overview/>

² <http://www.ebi.ac.uk/chebi/>

³ <http://www.w3.org/Submission/SWRL/>

only some of the problems related to the representation of structured objects, and our subsequent discussions with EBI researchers revealed the following drawbacks.

First, the DG approach does not allow one to define structures based on the *absence* of certain characteristics. For example, an inorganic molecule is commonly described as ‘a molecule *not* containing a carbon atom’, which can then be used to classify water as an inorganic molecule. Designing an axiomatisation that produces the desired entailment is very cumbersome with the DG approach: apart from stating that ‘each water molecule consists of one oxygen and two hydrogen atoms’, one must additionally state that ‘these three atoms are the only atoms in a water molecule’ and that ‘neither hydrogen nor oxygen atoms are carbon atoms’. Second, the separation conditions governing the interaction of the OWL 2 and DG components makes the combined language rather difficult to use, as no role can be used in both components. Third, the acyclicity condition from [12] is rather cumbersome: a modeller must add a number of negative class assertions to DGs so as to make any ontology with cyclic implications between DGs unsatisfiable. This solution fails to cleanly separate the semantic consequences of an ontology from the acyclicity check.

In response to this critique, in this paper we present a radically different approach to modelling complex objects via a novel formalism that we call Description Graph Logic Programs (DGLP). At the syntactic level, our approach combines DGs, rules, and OWL 2 RL axioms.⁴ In order to overcome the first problem, we give semantics to our formalism via a translation into logic programs interpreted under stable model semantics. As we show in Section 4, the resulting formalism can capture conditions based on the absence of information. Moreover, we address the second problem by ensuring decidability without the need for complex property separation conditions. To address the third problem, in Section 5 we discuss existing syntactic acyclicity conditions and argue that they unnecessarily rule out some very simple and intuitively reasonable ontologies. As a remedy, we present a novel *semantic acyclicity* condition. Roughly speaking, a precedence relation describing allowed implications between DGs is specified by the modeller; a cyclic ontology that is not compatible with this precedence relation entails a special propositional symbol. A cyclic ontology can still entail useful consequences, but termination of reasoning can no longer be guaranteed. In Section 6 we consider the problem of reasoning with negation-free ontologies and ontologies with stratified negation. We show that the standard bottom-up evaluation of logic programs can decide the relevant reasoning problems for semantically acyclic ontologies, and that it can also decide whether an ontology is semantically acyclic. In Section 7 we briefly discuss a preliminary evaluation of our formalism which indicates that reasoning with DGLP ontologies is practically feasible. Proofs of the technical results can be found online.⁵

2 Preliminaries

We assume the reader to be familiar with OWL and description logics; for brevity, we write OWL axioms using the DL notation. Let $\Sigma = (\Sigma_C, \Sigma_F, \Sigma_P)$ be a *first-order logic signature*, where Σ_C , Σ_F , and Σ_P are countably infinite sets of constant, function, and

⁴ <http://www.w3.org/TR/owl-profiles/>

⁵ <http://www.cs.ox.ac.uk/isg/people/despoina.magka/pubs/reports/DGLPTechnicalReport.pdf>

predicate symbols, respectively, and where Σ_P contains the 0-ary predicate \perp . The arity of a predicate A is given by $ar(A)$. A vector t_1, \dots, t_n of first-order terms is often abbreviated as \vec{t} . An *atom* is a first-order formula of the form $A(\vec{t})$, where $A \in \Sigma_P$ and \vec{t} is a vector of the terms $t_1, \dots, t_{ar(A)}$. A *rule* r is an implication of the form

$$B_1 \wedge \dots \wedge B_n \wedge \mathbf{not} B_{n+1} \wedge \dots \wedge \mathbf{not} B_m \rightarrow H_1 \wedge \dots \wedge H_\ell \quad (1)$$

where H_1, \dots, H_ℓ are atoms, B_1, \dots, B_m are atoms different from \perp , $m \geq 0$, and $\ell > 0$. Let $head(r) = \{H_i\}_{1 \leq i \leq \ell}$, $body^+(r) = \{B_i\}_{1 \leq i \leq n}$, $body^-(r) = \{B_i\}_{n < i \leq m}$, and $body(r) = body^+(r) \cup body^-(r)$. A rule r is *safe* if every variable that occurs in $head(r) \cup body^-(r)$ also occurs in $body^+(r)$. If $body(r) = \emptyset$ and r is safe, then r is a *fact*. We denote with $head_P(r)$, $body_P^+(r)$, $body_P^-(r)$, and $body_P(r)$ the set of predicates that occur in $head(r)$, $body^+(r)$, $body^-(r)$, and $body(r)$, respectively. A rule r is *function-free* if no function symbols occur in r . A *logic program* P is a set of rules. A logic program P is *negation-free* if, for each rule $r \in P$, we have $body^-(r) = \emptyset$.

Given a logic program P , $HU(P)$ (Herbrand Universe) is the set of all terms that can be formed using constants and functions from P (w.l.o.g. we assume that P contains at least one constant). If no variables occur in an atom (rule), then the atom (rule) is *ground*. Given a logic program P , the set $HB(P)$ is the set of all ground atoms constructed using the terms in $HU(P)$ and the predicates in P . The grounding of a rule r w.r.t. a set of terms T is the set of rules obtained by substituting the variables of r by the terms of T in all possible ways. Given a logic program P , the program $ground(P)$ is obtained from P by replacing each rule $r \in P$ with its grounding w.r.t. $HU(P)$.

Let $I \subseteq HB(P)$ be a set of ground atoms. Then, I *satisfies* a ground rule r if $body^+(r) \subseteq I$ and $body^-(r) \cap I = \emptyset$ imply $head(r) \subseteq I$. Furthermore, I is a model of a (not necessarily ground) program P , written $I \models P$, if $\perp \notin I$ and I satisfies each rule $r \in ground(P)$. Given a negation-free program P , set I is a *minimal model* of P if $I \models P$ and no $I' \subsetneq I$ exists such that $I' \models P$. The *Gelfond-Lifschitz reduct* P^I of a logic program P w.r.t. I is obtained from $ground(P)$ by removing each rule r such that $body^-(r) \cap I \neq \emptyset$, and removing all atoms $\mathbf{not} B_i$ in all the remaining rules. A set I is a *stable model* of P if I is a minimal model of P^I . Given a fact A , we write $P \models A$ if $A \in I$ for each stable model I of P ; otherwise, we write $P \not\models A$.

A substitution is a partial mapping of variables to ground terms. The result of applying a substitution θ to a term, atom, or a set of atoms M is written as $M\theta$ and is defined as usual. Let P be a logic program in which no predicate occurring in the head of a rule in P also occurs negated in the body of a (possibly different) rule in P . Operator T_P , applicable to a set of facts X , is defined as follows:

$$T_P(X) = X \cup \{h\theta \mid h \in head(r), r \in P, \theta \text{ maps the variables of } r \text{ to } HU(P \cup X) \text{ such that } body^+(r)\theta \subseteq X \text{ and } body^-(r)\theta \cap X = \emptyset\}$$

Let $T_P^0 = \emptyset$, let $T_P^i = T_P(T_P^{i-1})$ for $i \geq 1$, and let $T_P^\infty = \bigcup_{i=1}^\infty T_P^i$. Such P has at most one stable model, and T_P^∞ is the stable model of P if and only if $\perp \notin T_P^\infty$.

3 Motivating Application

We next motivate our work using examples from the chemical Semantic Web application mentioned in the introduction. The goal of this application is to automatically

classify chemical entities based on descriptions of their properties and structure. The inability of OWL to describe cyclic structures with sufficient precision causes problems when modelling chemical compounds, as molecules are highly cyclic. For example, the cyclobutane molecule contains four carbon atoms connected in a ring, as shown in Figure 1(a). One might try to represent this structure using the following OWL axiom:

$$\text{Cyclobutane} \sqsubseteq \text{Molecule} \sqcap (= 4 \text{ hasAtom.}[\text{Carbon} \sqcap (= 2 \text{ bond.} \text{Carbon})])$$

This axiom is satisfied in first-order interpretations I and I' shown in Figures 1(b) and 1(c), respectively; however, only interpretation I correctly reflects the structure of cyclobutane. Furthermore, interpretation I' cannot be ruled out by adding axioms due to the *tree-model property* of OWL, according to which each satisfiable TBox has at least one tree-shaped interpretation. This can prevent the entailment of certain desired consequences. For example, one cannot define the class of molecules containing four-membered rings that will be correctly identified as a superclass of cyclobutane.

The formalism from [12] addresses this problem by augmenting an OWL ontology with a set of rules and a set of *description graphs* (DGs), where each DG describes a complex object by means of a directed labeled graph. To avoid misunderstandings, we refer to the formalism from [12] as DGDL (Description Graph Description Logics), and to the formalism presented in this paper as DGLP (Description Graph Logic Programs). Thus, cyclobutane can be described using the DG shown in Figure 2(a). The first-order semantics of DGDL ontologies ensures that all models of an ontology correctly represent the DG structure; for example, interpretation I' from Figure 1(c) does not satisfy the DG in Figure 2(a). Nevertheless, the interpretation I'' shown in Figure 1(d) also satisfies the definition of cyclobutane under the semantics of DGDL ontologies. We next show how the presence of models with excess information can restrict entailments.

One might describe the class of hydrocarbon molecules (i.e., molecules consisting exclusively of hydrogens and carbons) using axiom (2). One would expect the definition of cyclobutane (as given in a DGDL ontology) and (2) to imply subsumption (3).

$$\text{Molecule} \sqcap \forall \text{hasAtom.}(\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon} \quad (2)$$

$$\text{Cyclobutane} \sqsubseteq \text{Hydrocarbon} \quad (3)$$

This, however, is not the case, since interpretation I'' does not satisfy axiom (3). One might preclude the existence of extra atoms by adding cardinality restrictions requiring each cyclobutane to have exactly four atoms. Even so, axiom (3) would not be entailed because of a model similar to I , but where one carbon atom is also an oxygen atom. One could eliminate such models by introducing disjointness axioms for all chemical elements. Such gradual circumscription of models, however, is not an adequate solution, as one can always think of additional information that needs to be ruled out.

In order to address such problems, we present a novel expressive formalism that we call Description Graph Logic Programs (DGLP). DGLP ontologies are similar to DGDL ontologies in that they extend OWL ontologies with DGs and rules. In our case, however, the ontology is restricted to OWL 2 RL so that the ontology can be translated into rules [6]. We give semantics to our formalism by translating DGLP ontologies into logic programs with function symbols. As is common in logic programming, the

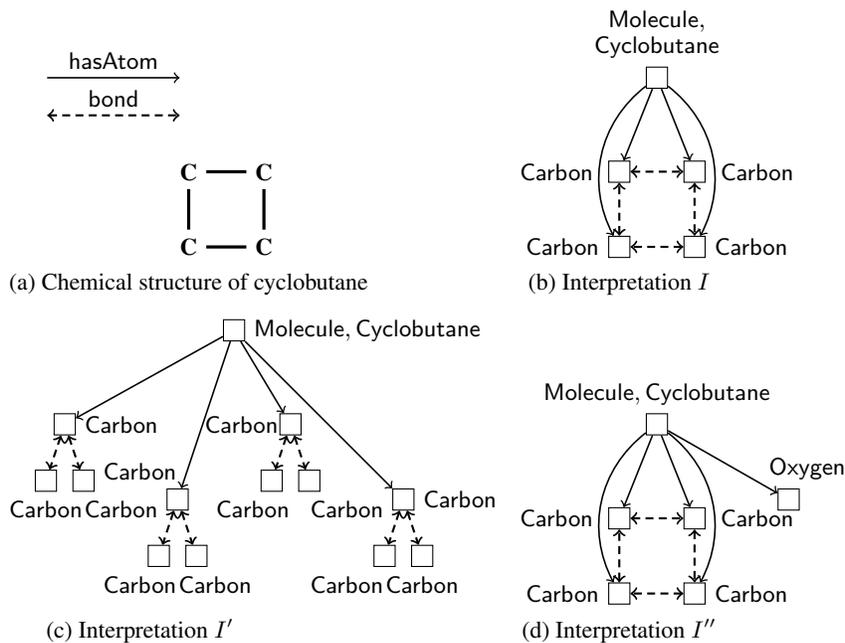


Fig. 1. The chemical structure and the models of cyclobutane

translation is interpreted under *stable* models. Consequently, interpretations such as I'' are not stable models of the DG in Figure 2(a), and hence subsumption (3) is entailed.

Logic programs with function symbols can axiomatise infinite non-tree-like structures, so reasoning with DGLP ontologies is trivially undecidable [2]. Our goal, however, is not to model arbitrarily large structures, but to describe complex objects up to a certain level of granularity. For example, acetic acid has a carboxyl part, and carboxyl has a hydroxyl part, but hydroxyl does not have an acetic acid part (see Fig. 3(a)). In Section 5 we exploit this intuition and present a new acyclicity condition that ensures decidability and allows for the modelling of naturally-arising molecular structures, such as acetic acid, that would be ruled out by existing syntactic acyclicity conditions [4, 11].

4 Description Graph Logic Programs

We now present the DGLP formalism in detail; we first define description graphs.

Definition 1 (Description Graph). A description graph $G = (V, E, \lambda, A, m)$ is a directed labeled graph where

- $V = \{1, \dots, n\}$ is a nonempty set of vertices,
- $E \subseteq V \times V$ is a set of edges,
- λ assigns a set of unary predicates $\lambda(v) \subseteq \Sigma_P$ to each vertex $v \in V$ and a set of binary predicates $\lambda(v_1, v_2) \subseteq \Sigma_P$ to each edge $(v_1, v_2) \in E$,

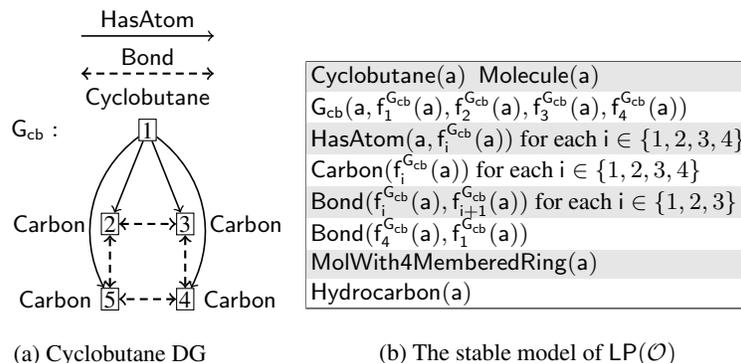


Fig. 2. Representing cyclobutane with DGLP

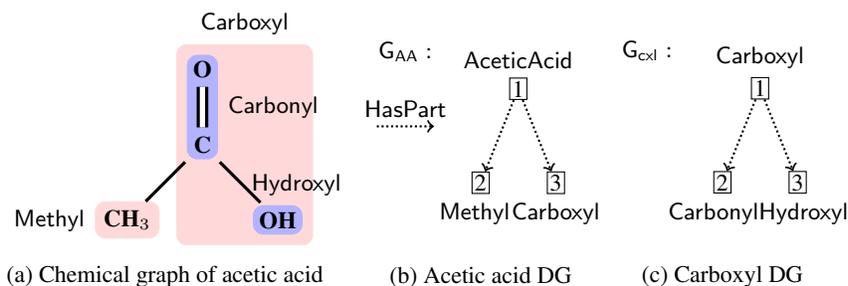


Fig. 3. The chemical graph of acetic acid and the G_{AA} and the G_{cxl} DGs

- $A \in \Sigma_P$ is a start predicate for G such that $A \in \lambda(1)$, and
- $m \in \{\Rightarrow, \Leftarrow, \Leftrightarrow\}$ is a mode for G .

A description graph (DG) abstracts the structure of a complex object by means of a directed labeled graph. For example, Figure 2 illustrates a DG that represents the structure of a cyclobutane molecule. The start predicate of the graph (Cyclobutane in this case) corresponds to the name of the object that the graph describes. The mode determines whether a graph should be interpreted as an ‘only if’, ‘if’, or ‘if and only if’ statement. More precisely, \Rightarrow means that each instance of the DG’s start predicate implies the existence of a corresponding instantiation of the entire graph structure; \Leftarrow means that an instantiation of a suitable graph structure is ‘recognised’ as an instance of the corresponding DG; and \Leftrightarrow means both of the above. Next we define *graph orderings*, which will play an important role in ensuring the decidability of DGLP.

Definition 2 (Graph Ordering). A graph ordering on a set of description graphs DG is a transitive and irreflexive relation $\prec \subseteq DG \times DG$.

Intuitively, a graph ordering specifies which DGs can imply the existence of instances of other DGs. For example, let G_{AA} be a graph that represents acetic acid, and

let G_{cxl} be a graph that represents the carboxyl group (Fig. 3); then, one might define \prec such that $G_{\text{AA}} \prec G_{\text{cxl}}$, so that an acetic acid instance may imply the existence of a carboxyl group instance, but not vice versa. We are now ready to define DGLP ontologies.

Definition 3 (DGLP Ontology). A DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F \rangle$ is a quadruple where DG is a finite set of description graphs, \prec is a graph ordering on DG , R is a finite set of function-free and safe rules, and F is a finite set of function-free facts.

For the sake of simplicity, we do not explicitly include an OWL 2 RL TBox into the definition of DGLP ontologies: OWL 2 RL axioms can be translated into rules as shown in [6] and included in R , and datatypes can be handled as in [10]. Similarly, we could think of F as an OWL 2 ABox, as ABox assertions correspond directly to facts [6]. An example of a DGLP ontology is $\{\{G_{\text{AA}}, G_{\text{cxl}}\}, \{(G_{\text{AA}}, G_{\text{cxl}})\}, \emptyset, \{\text{AceticAcid(a)}\}\}$.

We next define the semantics of DGLP via a translation into logic programs. Since R and F are already sets of rules and \prec serves only to check acyclicity, we only need to specify how to translate DGs into rules.

Definition 4 (Start, Layout, and Recognition Rule). Let $G = (V, E, A, \lambda, m)$ be a description graph and let $f_1^G, \dots, f_{|V|-1}^G$ be fresh distinct function symbols uniquely associated with G . The start rule s_G , the layout rule ℓ_G , and the recognition rule r_G of G are defined as follows (G is also used as a predicate of arity $|V|$):

$$A(x) \rightarrow G(x, f_1^G(x), \dots, f_{|V|-1}^G(x)) \quad (s_G)$$

$$G(x_1, \dots, x_{|V|}) \rightarrow \bigwedge_{i \in V, B \in \lambda(i)} B(x_i) \wedge \bigwedge_{\langle i, j \rangle \in E, R \in \lambda(i, j)} R(x_i, x_j) \quad (\ell_G)$$

$$\bigwedge_{i \in V, B \in \lambda(i), B \neq A} B(x_i) \wedge \bigwedge_{\langle i, j \rangle \in E, R \in \lambda(i, j)} R(x_i, x_j) \rightarrow G(x_1, \dots, x_{|V|}) \quad (r_G)$$

The start and layout rules of a description graph serve to unfold the graph's structure, whereas the recognition rule identifies instances of the start predicate. The function terms $f_1^G(x), \dots, f_{|V|-1}^G(x)$ correspond to existential restrictions whose existentially quantified variables have been skolemised.

Example 1. The start rule and layout rule that correspond to the DG of cyclobutane from Figure 2 (assuming mode \Rightarrow as specified in Definition 5) are the following.

$$\text{Cyclobutane}(x) \rightarrow G_{\text{cb}}(x, f_1^{\text{Gcb}}(x), f_2^{\text{Gcb}}(x), f_3^{\text{Gcb}}(x), f_4^{\text{Gcb}}(x)) \quad (s_{G_{\text{cb}}})$$

$$G_{\text{cb}}(x_1, x_2, x_3, x_4, x_5) \rightarrow \text{Cyclobutane}(x_1) \wedge \bigwedge_{2 \leq i \leq 4} \text{Bond}(x_i, x_{i+1}) \wedge \text{Bond}(x_5, x_2) \wedge \bigwedge_{2 \leq i \leq 5} \text{HasAtom}(x_1, x_i) \wedge \bigwedge_{2 \leq i \leq 5} \text{Carbon}(x_i) \quad (\ell_{G_{\text{cb}}})$$

Next, we define $\text{Axioms}(DG)$, which is a logic program that encodes a set of DGs.

Definition 5 (Axioms(DG)). For a description graph $G = (V, E, \lambda, A, m)$, the program $\text{Axioms}(G)$ is the set of rules that contains the start rule s_G and the layout rule ℓ_G if $m \in \{\Rightarrow, \Leftrightarrow\}$, and the recognition rule r_G if $m \in \{\Leftarrow, \Leftrightarrow\}$. For a set of description graphs $DG = \{G_i\}_{1 \leq i \leq n}$, let $\text{Axioms}(DG) = \bigcup_{G_i \in DG} \text{Axioms}(G_i)$.

For each DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F \rangle$, we denote with $\text{LP}(\mathcal{O})$ the program $\text{Axioms}(DG) \cup R \cup F$. We check whether D subsumes C as in standard OWL reasoning: we assert $C(a)$ for a a fresh individual, and we check whether $D(a)$ is entailed.

Definition 6 (Subsumption). *Let \mathcal{O} be a DGLP ontology, let C and D be unary predicates occurring in \mathcal{O} , and let a be a fresh constant not occurring in \mathcal{O} . Then, D subsumes C w.r.t. \mathcal{O} , written $\mathcal{O} \models C \sqsubseteq D$, if $\text{LP}(\mathcal{O}) \cup \{C(a)\} \models D(a)$ holds.*

Example 2. We now show how a DGLP ontology can be used to obtain the inferences described in Section 3. Rule (r_1) encodes the class of four-membered ring molecules and rules (r_2) and (r_3) represent the class of hydrocarbons.

$$\begin{aligned} & \text{Molecule}(x) \wedge \bigwedge_{1 \leq i \leq 4} \text{HasAtom}(x, y_i) \wedge \bigwedge_{1 \leq i \leq 3} \text{Bond}(y_i, y_{i+1}) \wedge \text{Bond}(y_4, y_1) \\ & \bigwedge_{1 \leq i < j \leq 4} \text{not } y_i = y_j \rightarrow \text{MolWith4MemberedRing}(x) \quad (r_1) \\ & \text{Molecule}(x) \wedge \text{HasAtom}(x, y) \wedge \text{not Carbon}(y) \wedge \text{not Hydrogen}(y) \rightarrow \text{NHC}(x) \quad (r_2) \\ & \text{Molecule}(x) \wedge \text{not NHC}(x) \rightarrow \text{HydroCarbon}(x) \quad (r_3) \\ & \text{Cyclobutane}(x) \rightarrow \text{Molecule}(x) \quad (r_4) \end{aligned}$$

The use of the equality predicate $=$ in the body of r_1 does not require an extension to our syntax: if $=$ occurs only in the body and not in the head of the rules, then negation of equality can be implemented using a built-in predicate. We also state that cyclobutane is a molecule using (r_4) that corresponds to the OWL 2 RL axiom $\text{Cyclobutane} \sqsubseteq \text{Molecule}$. Let $DG = \{\text{G}_{\text{cb}}\}$, let $\prec = \emptyset$, let $R = \{r_i\}_{i=1}^4$, let $F = \{\text{Cyclobutane}(a)\}$, and let $\mathcal{O} = \langle DG, \prec, R, F \rangle$. Figure 2(b) shows the only stable model of $\text{LP}(\mathcal{O})$ by inspection of which we see that $\text{LP}(\mathcal{O}) \models \text{Hydrocarbon}(a)$ and $\text{LP}(\mathcal{O}) \models \text{MolWith4MemberedRing}(a)$, as expected.

5 Semantic Acyclicity

Reasoning about logic programs with function symbols is undecidable in general [2]. This problem is similar to reasoning about datalog programs with existentially quantified rule heads (known as tuple-generating dependencies or tgds) [3]. For such programs, conditions such as weak acyclicity [4] or super-weak acyclicity [11] ensure the termination of bottom-up reasoning algorithms: these conditions examine the syntactic structure of the rules and check whether values created by a rule's head can be propagated so as to eventually satisfy the premise of the same rule. Such conditions can also be applied to DGLP ontologies; however, they may overestimate the propagation of values introduced by existential quantification and thus rule out unproblematical programs that generate only finite structures. As shown in Example 4, this is the case for programs that naturally arise from DGLP representations of molecular structures.

To mitigate this problem, we propose a new *semantic* acyclicity condition. The idea is to detect repetitive construction of DG instances by checking the entailment of a special propositional symbol Cycle . In particular, the graph ordering \prec of a DGLP ontology

\mathcal{O} is used to extend $\text{LP}(\mathcal{O})$ with rules that derive Cycle whenever an instance of a DG G_1 implies existence of an instance of a DG G_2 but $G_1 \not\prec G_2$.

Definition 7 (Check(\mathcal{O})). Let $G_i = (V_i, E_i, \lambda_i, A_i, m_i)$, $i \in \{1, 2\}$ be two description graphs. We define $\text{ChkPair}(G_1, G_2)$ and $\text{ChkSelf}(G_i)$ as follows:

$$\text{ChkPair}(G_1, G_2) = \{G_1(x_1, \dots, x_{|V_1|}) \wedge A_2(x_k) \rightarrow \text{Cycle} \mid 1 \leq k \leq |V_1|\} \quad (4)$$

$$\text{ChkSelf}(G_i) = \{G_i(x_1, \dots, x_{|V_i|}) \wedge A_i(x_k) \rightarrow \text{Cycle} \mid 1 < k \leq |V_i|\} \quad (5)$$

Let $DG = \{G_i\}_{1 \leq i \leq n}$ be a set of description graphs and let \prec be a graph ordering on DG . We define $\text{Check}(DG, \prec)$ as follows:

$$\text{Check}(DG, \prec) = \bigcup_{i, j \in \{1, \dots, n\}, i \neq j, G_i \not\prec G_j} \text{ChkPair}(G_i, G_j) \cup \bigcup_{1 \leq i \leq n} \text{ChkSelf}(G_i)$$

For a DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F \rangle$, we define $\text{Check}(\mathcal{O}) = \text{Check}(DG, \prec)$.

Example 3. Figure 3(a) shows the structure of acetic acid molecules and the parts they consist of. In this example, however, we focus on the description graphs for acetic acid (G_{AA}) and carboxyl (G_{cxl}), which are shown in Figures 3(b) and 3(c), respectively. Since an instance of acetic acid implies the existence of an instance of a carboxyl, but not vice versa, we define our ordering as $G_{AA} \prec G_{cxl}$. Thus, for $DG = \{G_{AA}, G_{cxl}\}$ and $\prec = \{(G_{AA}, G_{cxl})\}$, set $\text{Check}(DG, \prec)$ contains the following rules:

$$G_{cxl}(x_1, x_2, x_3) \wedge \text{AceticAcid}(x_i) \rightarrow \text{Cycle} \quad \text{for } 1 \leq i \leq 3$$

$$G_{AA}(x_1, x_2, x_3) \wedge \text{AceticAcid}(x_i) \rightarrow \text{Cycle} \quad \text{for } 2 \leq i \leq 3$$

$$G_{cxl}(x_1, x_2, x_3) \wedge \text{Carboxyl}(x_i) \rightarrow \text{Cycle} \quad \text{for } 2 \leq i \leq 3$$

Definition 8. A DGLP ontology \mathcal{O} is said to be semantically acyclic if and only if $\text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O}) \not\models \text{Cycle}$.

Example 4. Let $DG = \{G_{AA}, G_{cxl}\}$ with $m_{AA} = m_{cxl} = \Leftrightarrow$, let $\prec = \{(G_{AA}, G_{cxl})\}$, let $F = \{\text{AceticAcid}(a)\}$, and let $\mathcal{O} = \langle DG, \prec, \emptyset, F \rangle$. The logic program $\text{LP}(\mathcal{O})$ contains F and the following rules (HP abbreviates HasPart):

$$\text{AceticAcid}(x) \rightarrow G_{AA}(x, f_1(x), f_2(x))$$

$$G_{AA}(x, y, z) \rightarrow \text{AceticAcid}(x) \wedge \text{Methyl}(y) \wedge \text{Carboxyl}(z) \wedge \text{HP}(x, y) \wedge \text{HP}(x, z)$$

$$\text{Methyl}(y) \wedge \text{Carboxyl}(z) \wedge \text{HP}(x, y) \wedge \text{HP}(x, z) \rightarrow G_{AA}(x, y, z)$$

$$\text{Carboxyl}(x) \rightarrow G_{cxl}(x, g_1(x), g_2(x))$$

$$G_{cxl}(x, y, z) \rightarrow \text{Carboxyl}(x) \wedge \text{Carbonyl}(y) \wedge \text{Hydroxyl}(z) \wedge \text{HP}(x, y) \wedge \text{HP}(x, z)$$

$$\text{Carbonyl}(y) \wedge \text{Hydroxyl}(z) \wedge \text{HP}(x, y) \wedge \text{HP}(x, z) \rightarrow G_{cxl}(x, y, z)$$

Let also $\text{Check}(\mathcal{O}) = \text{Check}(DG, \prec)$ as defined in Example 3. We can easily compute the stable model of $P = \text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O})$ using the T_P operator and check that Cycle is not in the (only) stable model of P ; so $P \not\models \text{Cycle}$ and \mathcal{O} is semantically acyclic. However, P is neither weakly [4] nor super-weakly acyclic [11]. This, we believe, justifies the importance of semantic acyclicity for our applications.

6 Reasoning with DGLP Ontologies

Initially, we consider the problem of reasoning with a negation-free DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F \rangle$. Intuitively, one can apply the T_P operator to $P = \text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O})$ and compute T_P^1, \dots, T_P^i and so on. By Theorem 5, for some i we will either reach a fixpoint or derive Cycle. In the former case, we have the stable model of \mathcal{O} (if $\perp \notin T_P^i$), which we can use to decide the relevant reasoning problems; in the latter case, we know that \mathcal{O} is not acyclic.

Theorem 5. *Let $\mathcal{O} = \langle DG, \prec, R, F \rangle$ be a DGLP ontology with R negation-free, and let $P = \text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O})$. Then, $\text{Cycle} \in T_P^i$ or $T_P^{i+1} = T_P^i$ for some $i \geq 1$.*

Checking the semantic acyclicity of \mathcal{O} is thus decidable. If the stable model of $\text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O})$ is infinite, then Cycle is derived; note that the inverse does not hold as semantic acyclicity is a sufficient but not necessary termination condition.

Next, we consider the case of DGLP ontologies with stratified negation-as-failure. We start by recapitulating several definitions. For each program P , a *stratification* of P is a mapping $\sigma : P \rightarrow \mathbb{N}$ such that for each rule $r \in P$ we have (i) if $B \in \text{body}_P^+(r)$, then $\sigma(r') \leq \sigma(r)$ for each $r' \in P$ with $B \in \text{head}_P(r')$ and (ii) if $B \in \text{body}_P^-(r)$, then $\sigma(r') < \sigma(r)$ for each $r' \in P$ with $B \in \text{head}_P(r')$. A logic program P is *stratifiable* if there exists a stratification of P . Moreover, a partition P_1, \dots, P_n of P is a *stratification partition* of P w.r.t. σ if, for each $r \in P$, we have $r \in P_{\sigma(r)}$. The sets P_1, \dots, P_n are called the *strata* of P . Let $U_{P_0}^\infty = T_{P_1}^1, U_{P_j}^i = T_{P_j}^i(U_{P_{j-1}}^\infty)$ for $1 \leq j \leq n$ and $i \geq 1$ and $U_{P_j}^\infty = T_{P_j}^\infty(U_{P_{j-1}}^\infty)$. The stable model of P is given by $U_{P_n}^\infty$. Next we introduce the notion of a DG-stratification, which ensures that the cycle detection rules are assigned to the strata containing the relevant start rules of DGs.

Definition 9 (DG-stratification). *Let $\mathcal{O} = \langle DG, \prec, R, F \rangle$ be a DGLP ontology, let $P = \text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O})$ and let P_1, \dots, P_n be a stratification partition of P w.r.t. some stratification σ of P . Then, σ is a DG-stratification if*

- for each $G_1, G_2 \in DG$ such that $G_1 \neq G_2$, $G_1 \not\prec G_2$, and $\{s_{G_1}, s_{G_2}\} \subseteq P_i$, we have $\text{ChkPair}(G_1, G_2) \subseteq P_i$, and
- for each $G \in DG$ such that $s_G \in P_i$, we have $\text{ChkSelf}(G) \subseteq P_i$.

The following result shows that, as long as $\text{LP}(\mathcal{O})$ is stratified, one can always assign the cycle checking rules in $\text{Check}(\mathcal{O})$ to the appropriate strata and thus obtain a DG-stratification of $\text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O})$.

Lemma 1. *Let $\mathcal{O} = \langle DG, \prec, R, F \rangle$ be a DGLP ontology. If σ is a stratification of $\text{LP}(\mathcal{O})$, then σ can be extended to a DG-stratification σ' of $\text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O})$.*

The following theorem implies that, given a stratifiable DGLP ontology, we can decide whether the ontology is semantically acyclic, and if so, we can compute its stable model and thus solve all relevant reasoning problems; please note that we show the decidability of semantic acyclicity only for ontologies with stratified negation.

Theorem 6. *Let \mathcal{O} be a DGLP ontology and $P = \text{LP}(\mathcal{O}) \cup \text{Check}(\mathcal{O})$. If P_1, \dots, P_n is a stratification partition of P w.r.t. a DG-stratification of P , then, for each j with $1 \leq j \leq n$, there exists $i \geq 1$ such that $\text{Cycle} \in U_{P_j}^i$, or $U_{P_j}^{i+1} = U_{P_j}^i$ and $U_{P_j}^i$ is finite.*

7 Implementation Results and Discussion

In order to test the applicability of our approach in practice, we developed a prototypical implementation based on the XSB system.⁶ Using data extracted from ChEBI, we built a number of DGLP ontologies with stratified NAF and we checked each ontology for acyclicity and for entailed subsumptions: all ontologies were found acyclic and all molecules were classified as expected; additionally, testing subsumptions for an ontology representing 70 molecules did not require more than a few minutes on a standard desktop computer. Given the prototypical nature of our application, we consider the results as evidence of the practical feasibility of our approach.

In this paper we have laid the theoretical foundations of a novel, expressive, and OWL 2 RL-compatible ontology language that is well suited to modelling objects with complex structure. In the future, we plan to modify our approach in order to avoid the explicit definition of graph ordering by the modeller; furthermore, we shall investigate whether semantic acyclicity can be combined with other conditions (such as those defined in [1]) in order to obtain a more general acyclicity check. Finally, we will optimise our prototype in order to obtain a fully-scalable chemical classification system.

References

1. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: walking the decidability line. *Artif. Intell.* 175(9-10), 1620–1654 (2011)
2. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: *Proc. ICALP*. pp. 73–85 (1981)
3. Cali, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: *LICS* (2010)
4. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. In: *ICDT*. pp. 207–224 (2003)
5. Graves, H.: Representing product designs using a description graph extension to OWL 2. In: *Proc. of the 5th OWLED Workshop* (2009)
6. Grosz, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: *WWW* (2003)
7. Hastings, J., Dumontier, M., Hull, D., Horridge, M., Steinbeck, C., Sattler, U., Stevens, R., Hörne, T., Britz, K.: Representing chemicals using OWL, description graphs and rules. In: *OWLED* (2010)
8. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: tractable rules for OWL 2. In: *Proc. of the 7th Int. Semantic Web Conference (ISWC 2008)*. pp. 649–664 (2008)
9. Levy, A.Y., Rousset, M.C.: Combining horn rules and description logics in CARIN. *Artificial Intelligence* 104(1–2), 165–209 (1998)
10. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. *J. of Artificial Intelligence Research* 23, 667–726 (2004)
11. Marnette, B.: Generalized schema-mappings: from termination to tractability. In: *PODS* (2009)
12. Motik, B., Grau, B.C., Horrocks, I., Sattler, U.: Representing ontologies using description logics, description graphs, and rules. *Artif. Int.* 173, 1275–1309 (2009)
13. Rector, A.L., Nowlan, W.A., Glowinski, A.: Goals for concept representation in the GALEN project. In: *SCAMC '93*. pp. 414–418 (1993)

⁶ <http://xsb.sourceforge.net/>

Axiom Pinpointing Using an Assumption-Based Truth Maintenance System

Hai Nguyen, Natasha Alechina, and Brian Logan

University of Nottingham

1 Introduction

The problem of *axiom pinpointing* [1, 22], that is, finding the minimal set of axioms responsible for an unwanted consequence, is an important problem in ontology debugging. One approach to identifying the axioms responsible for an unwanted consequence is to trace dependencies between inferences leading to the consequence. Several authors have proposed *truth maintenance systems* as a means of keeping track of dependencies or inferences in ontologies, e.g., [21, 5, 7]. In this paper we show that truth maintenance systems can also be used for axiom pinpointing. More specifically, we present a system which returns all minimal sets of axioms responsible for the derivation of inconsistency in an unfoldable \mathcal{ALC} ontology. Following Sirin *et al* [24], we refer to these sets of axioms as *explanations*.

Our approach involves using a modified Assumption-Based Truth Maintenance System (ATMS) [11] to trace inferential dependencies between formulas and compute the minimal sets of ontology axioms responsible for a contradiction. The main technical contribution of the paper is extending the ATMS to deal with disjunctions. We generalise the notion of an ATMS environment (a set of axioms from which a formula is derivable) to include the non-deterministic choices required for the derivation of the formula. We show that this extended ATMS (which we call the D-ATMS), combined with a tableau reasoner, produces correct, complete and minimal explanations for a contradiction in an unfoldable \mathcal{ALC} ontology. We have developed a prototype implementation of our approach which we call AOD. Preliminary results of experiments comparing AOD, MUPSter and the Pellet explanation service are encouraging, and suggest that AOD can outperform MUPSter and Pellet on both synthetic and real-world ontologies.

2 The Reasoner

Our ontology debugging framework, AOD, consists of two components: a tableau reasoner, and the D-ATMS (described in Section 3).

The reasoner takes a TBox as an input. To check for incoherence, we check whether a contradiction is derivable from the TBox and a statement of non-emptiness of a concept, eg $A(a)$. We refer to all TBox and ABox elements as formulas and reserve the term ‘axiom’ for the input formulas.

The reasoner is a tableau reasoner for \mathcal{ALC} with unfoldable TBoxes [17, 2], and uses essentially the same rules as in [22, 16]:

- \sqsubseteq -rule from $A(a)$ and $A \sqsubseteq C$ derive $C(a)$
- \sqcap -rule from $(C_1 \sqcap \dots \sqcap C_n)(a)$ derive $C_1(a), \dots, C_n(a)$
- \exists -rule from $(\exists s.C)(a)$ derive $s(a, b), C(b)$ where b is a new individual and $(\exists s.C)(a)$ has not been used before to generate another new individual
- \forall -rule from $(\forall s.C)(a)$ and $s(a, b)$ derive $C(b)$
- \perp -rule from $A(a)$ and $\neg A(a)$ derive \perp
- \sqcup -rule from $(C_1 \sqcup \dots \sqcup C_n)(a)$, derive choices $C_1(a), \dots, C_n(a)$

where A is an atomic concept, C and D are arbitrary concepts, a, b are constants, and s is a role.

The \sqcup -rule creates branches in the tableaux for each disjunct (choice) $C_1(a), \dots, C_n(a)$. A tableau is a tree where nodes are sets of formulas, and children of a node are obtained by applying inference rules to formulas in the node, so that the child node(s) contains all the formulas from the parent node and the newly derived formula. For readability, we will sometimes show only the new formula in a child node, with the understanding that all the formulas higher up on the branch belong to the node as well. If a node contains several disjunctions, for example $B_1 \sqcup B_2 \sqcup B_3(a)$ and $C_1 \sqcup C_2(b)$ as in Figure 1, the order in which the disjunction rule is applied does not matter, but once this order is fixed, the choices for the second disjunction are repeated under each of the choices for the first disjunction:

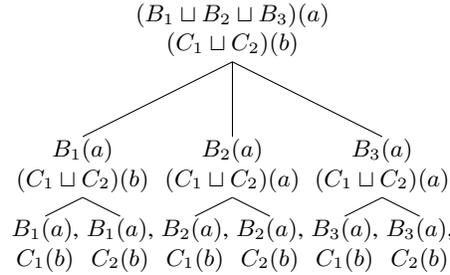


Fig. 1: Tableau with nested disjunctions

The reasoner derives consequences by applying inference rules to axioms and previously derived formulas. An *inference* $\phi_1, \dots, \phi_n \xrightarrow{r} \phi$ indicates that the formula ϕ can be derived from the set of formulas ϕ_1, \dots, ϕ_n using the inference rule r . The reasoner does not stop after a contradiction is derived on a branch, but continues to apply inference rules until no new rule applications are possible. A rule application is new if the inference rule has not been used before with the same premises. The reasoner never repeats the same rule application.

3 The D-ATMS

The D-ATMS maintains dependencies between formulas inferred by the reasoner. To do so, the D-ATMS builds and maintains a *justification graph*. Each node in the graph corresponds to a formula or a justification. We denote the node corresponding to a formula

ϕ by n_ϕ . Axioms are represented by *axiom nodes*, and inconsistency is represented by a distinguished *false node*, n_\perp . A *justification* is a structure $j : n_{\phi_1}, \dots, n_{\phi_k} \Rightarrow n_\phi$, where $n_{\phi_1}, \dots, n_{\phi_k}$ are nodes corresponding to the antecedents of an inference rule application, n_ϕ is a node corresponding to the consequent, and j is the justification id, a unique, sequentially assigned integer that identifies the justification.¹ In the interests of readability, we will often refer to a formula node n_ϕ by the formula ϕ it represents.

When the reasoner applies an inference rule, it passes the resulting inference to the D-ATMS, causing the D-ATMS to update the justification graph. The reasoner keeps making inferences until no new inferences can be made. The D-ATMS is then invoked to compute all explanations for \perp . An *explanation* consists of all minimal sets of axioms from which \perp can be derived, and, optionally, the sequence of inference rules necessary to derive \perp from each set of axioms. The explanations returned by the D-ATMS are guaranteed to be correct (in the sense that \perp is derivable from each of the returned sets of axioms) and minimal (in the sense that \perp is not derivable from their proper subsets).

As an example, consider the following TBox inspired by the MadCow example from the OilEd tutorial:

- ax1** $Sheep \sqsubseteq Animal$
- ax2** $Cow \sqsubseteq Animal \sqcap \forall eats. \neg Animal$
- ax3** $MadCow \sqsubseteq Cow \sqcap \exists eats. (Sheep \sqcup Cow)$

we also add the assumption $MadCow(a)$. The inferences made by the reasoner give rise to the following justifications (note that $Animal(b)$ has two justifications):

- j_1 $MadCow(a), MadCow \sqsubseteq Cow \sqcap \exists eats. (Sheep \sqcup Cow) \Rightarrow Cow \sqcap \exists eats. (Sheep \sqcup Cow)(a)$
- j_2 $Cow \sqcap \exists eats. (Sheep \sqcup Cow)(a) \Rightarrow Cow(a)$
- j_3 $Cow \sqcap \exists eats. (Sheep \sqcup Cow)(a) \Rightarrow \exists eats. (Sheep \sqcup Cow)(a)$
- j_4 $Cow(a), Cow \sqsubseteq Animal \sqcap \forall eats. \neg Animal, \Rightarrow Animal \sqcap \forall eats. \neg Animal(a)$
- j_5 $\exists eats. (Sheep \sqcup Cow)(a) \Rightarrow eats(a, b)$
- j_6 $\exists eats. (Sheep \sqcup Cow)(a) \Rightarrow (Sheep \sqcup Cow)(a)$
- j_7 $(Animal \sqcap \forall eats. \neg Animal)(a) \Rightarrow Animal(a)$
- j_8 $(Animal \sqcap \forall eats. \neg Animal)(a) \Rightarrow \forall eats. \neg Animal(a)$
- j_9 $eats(a, b), \forall eats. \neg Animal(a) \Rightarrow \neg Animal(b)$
- j_{10} $(Sheep \sqcup Cow)(a) \Rightarrow Sheep(a)$ (non-deterministic)
- j_{11} $(Sheep \sqcup Cow)(a) \Rightarrow Cow(a)$ (non-deterministic)
- j_{12} $Sheep(b), Sheep \sqsubseteq Animal \Rightarrow Animal(b)$
- j_{13} $Animal(b), \neg Animal(b) \Rightarrow \perp$
- j_{14} $Cow(b), Cow \sqsubseteq Animal \sqcap \forall eats. \neg Animal \Rightarrow (Animal \sqcap \forall eats. \neg Animal)(b)$
- j_{15} $(Animal \sqcap \forall eats. \neg Animal)(b) \Rightarrow Animal(b)$
- j_{16} $(Animal \sqcap \forall eats. \neg Animal)(b) \Rightarrow \forall eats. \neg Animal(b)$

and the justification graph is shown in Figure 2.

In a standard ATMS [11], each node has a *label* consisting of a set of *environments*. An environment is a minimal set of axioms from which the corresponding formula is

¹ Note that we use the term justification as it is used in ATMS literature, rather than to mean the minimal set of axioms responsible for an entailment as in, e.g., [4].

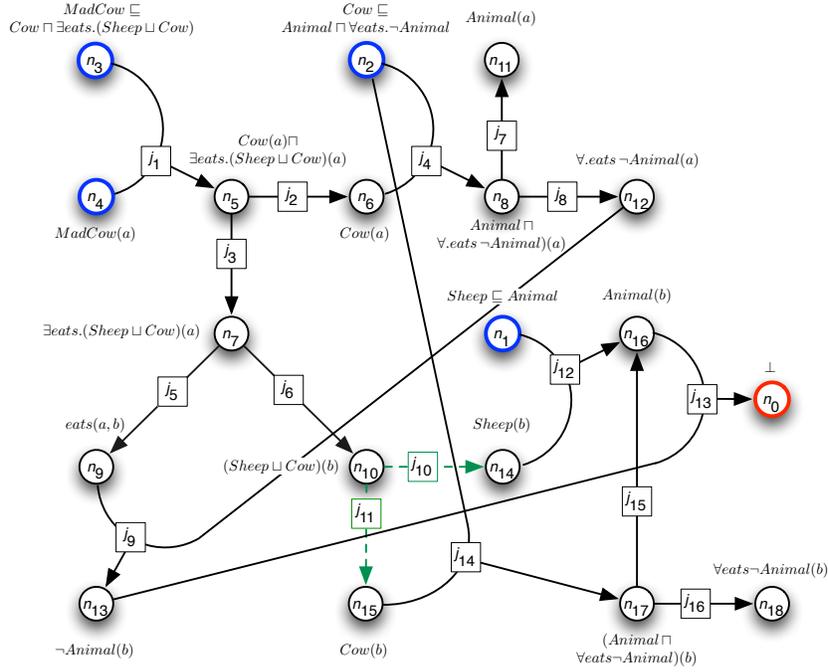


Fig. 2: Justification graph. Formula nodes are round, axioms are blue, \perp is red. Justification nodes are square, non-deterministic justifications are green with dashed arrows.

derivable (an explanation). For example, $Animal(a)$ in Figure 2 would have an environment $\{Cow \sqsubseteq Animal \sqcap \forall eats.\neg Animal, MadCow \sqsubseteq Cow \sqcap \exists eats.(Sheep \sqcup Cow), MadCow(a)\}$. Labels are computed and minimised at the same time as the justification graph is built. In contrast, in AOD, labels are computed only for the nodes which belong to the part of the justification graph which is involved in the derivation of \perp (is reachable from \perp following the edges backwards), and only after the graph is complete. In our example, the relevant part is the graph without the justifications j_7, j_{16} and the nodes n_{11}, n_{18} .

4 Computing Labels

In this section, we explain how the standard ATMS label computation algorithms are generalised to deal with disjunctions. Basically, the generalisation consists in keeping track of dependencies on disjunctive choices in addition to dependencies on axioms.

As in a standard ATMS, each node in the justification graph has a *label* consisting of a set of environments. However in the D-ATMS, an environment represents a set of axioms and choices under which a particular formula holds.

Definition 1 (environment). An environment e is a pair $(\mathcal{A}, \mathcal{C})$ where \mathcal{A} is a set of axioms and \mathcal{C} is a sequence of choice sets $[c_1, \dots, c_k]$ of length $k \geq 0$. Each choice set

c_i is a pair (d_i, b_i) where $d_i = \psi_1 \sqcup \dots \sqcup \psi_n$ is a disjunction and $b_i \subset \{\psi_1, \dots, \psi_n\}$ is a set of choices for d_i (i.e., a subset of the disjuncts appearing in the disjunction).

The presence of an environment $(\mathcal{A}, \mathcal{C})$ in the label of a node n_ϕ indicates that ϕ can be derived from the axioms \mathcal{A} together with a sequence of choices from \mathcal{C} . The choice sequence corresponds to a (set of) tableau branch(es): each *choice* consists of a disjunction d_i and one or more of the disjuncts appearing in d_i . If ϕ can be derived from all the disjuncts appearing in d_i , we have eliminated dependency on all choices for d_i , and the choice set for d_i can be removed from \mathcal{C} . If the sequence of choice sets is empty, then ϕ does not depend on any choices (i.e., it can be derived from only the axioms \mathcal{A}). For example, the presence of the environment $(\{\phi_1, \dots, \phi_k\}, [])$ in the label of a node n_ϕ means that ϕ has been derived by the reasoner from the axioms ϕ_1, \dots, ϕ_k .

Environments in the D-ATMS thus capture the branching structure of a tableau. For example, in the tableau in Figure 1 an environment for $B_1(a)$ will have a choice sequence $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a))]$ and $C_1(b)$ will have a choice sequence $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a)), ((C_1 \sqcup C_2)(b), C_1(b))]$. The order of choice sets in a choice sequence comes from the order in which the \sqcup -rule is applied to disjunctions on the corresponding branch. If one choice sequence corresponds to a prefix of another, then the first choice sequence depends on fewer disjunctive choices. This intuition may be helpful when considering the definition of subsumption for environments below.

The *label* of a node contains the set of environments from which the formula corresponding to the node can be derived. The label of n_\perp consists of a set of inconsistent environments or nogoods.

To define the D-ATMS algorithms for computing labels, we need the following primitive operations on environments and labels which generalise and extend the corresponding notions in [11].

We say that a choice sequence \mathcal{C}_1 is a prefix of a choice sequence \mathcal{C}_2 , $\mathcal{C}_1 \preceq \mathcal{C}_2$, if $\mathcal{C}_1 = [(d_1, b_1), \dots, (d_k, b_k)]$ and $\mathcal{C}_2 = [(d'_1, b'_1), \dots, (d'_n, b'_n)]$, $k \leq n$ and for every $i \leq k$, $d_i = d'_i$ and $b'_i \subseteq b_i$. $\mathcal{C}_1 \prec \mathcal{C}_2$ iff $\mathcal{C}_1 \preceq \mathcal{C}_2$ and $\mathcal{C}_2 \not\preceq \mathcal{C}_1$.

Definition 2 (Subsumption of environments). An environment $(\mathcal{A}_1, \mathcal{C}_1)$ subsumes an environment $(\mathcal{A}_2, \mathcal{C}_2)$, $(\mathcal{A}_1, \mathcal{C}_1) \subseteq_s (\mathcal{A}_2, \mathcal{C}_2)$ iff $\mathcal{A}_1 \subseteq \mathcal{A}_2$, and $\mathcal{C}_1 \preceq \mathcal{C}_2$. $(\mathcal{A}_1, \mathcal{C}_1) \subset_s (\mathcal{A}_2, \mathcal{C}_2)$ iff $(\mathcal{A}_1, \mathcal{C}_1) \subseteq_s (\mathcal{A}_2, \mathcal{C}_2)$ and $(\mathcal{A}_2, \mathcal{C}_2) \not\subseteq_s (\mathcal{A}_1, \mathcal{C}_1)$.

An environment e is *nogood* if it is subsumed by an environment in the label of the false node n_\perp .

Definition 3 (Union of environments). The union of two environments $e_1 = (\mathcal{A}_1, \mathcal{C}_1)$ and $e_2 = (\mathcal{A}_2, \mathcal{C}_2)$, $e_1 \cup_{\leq} e_2 = (\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{C}_1 \cup_{\leq} \mathcal{C}_2)$ if \mathcal{C}_1 and \mathcal{C}_2 are sequences of choice sets for which $\mathcal{C}_1 \cup_{\leq} \mathcal{C}_2$ is defined, otherwise $e_1 \cup_{\leq} e_2$ is not defined. \cup_{\leq} for sequences of choice sets is defined as follows:

1. if $\mathcal{C}_1 \preceq \mathcal{C}_2$ then $\mathcal{C}_1 \cup_{\leq} \mathcal{C}_2 = \mathcal{C}_2$;
2. if $\mathcal{C}_2 \preceq \mathcal{C}_1$ then $\mathcal{C}_1 \cup_{\leq} \mathcal{C}_2 = \mathcal{C}_1$;
3. for all other cases, $\mathcal{C}_1 \cup_{\leq} \mathcal{C}_2$ is not defined.

Intuitively, environments of two antecedents can be combined by \cup_{\leq} to form an environment of the consequent if the antecedents belong to the same branch of the tableau.

Definition 4 (Merge of environments). *The merge of two environments $e_1 = (\mathcal{A}_1, \mathcal{C}_1)$ and $e_2 = (\mathcal{A}_2, \mathcal{C}_2)$, $e_1 \cup_+ e_2 = (\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{C}_1 \cup_+ \mathcal{C}_2)$ if \mathcal{C}_1 and \mathcal{C}_2 are sequences of choice sets for which \cup_+ is defined. Otherwise, $e_1 \cup_+ e_2$ is not defined. \cup_+ for sequences of choice sets is defined as follows:*

1. if $\mathcal{C}_1 = [(d_1, b_1), \dots, (d_n, b_n)]$ and $\mathcal{C}_2 = [(d'_1, b'_1), \dots, (d'_n, b'_n)]$, $n \geq 1$, and for every $i < n$ $d_i = d'_i$ and $b'_i = b_i$ (in other words, \mathcal{C}_1 and \mathcal{C}_2 are the same apart from their last element), $d_n = d'_n$, $b_n \neq b'_n$, then
 - (a) if $b_n \cup b'_n$ does not include all the disjuncts in d_n , then $\mathcal{C}_1 \cup_+ \mathcal{C}_2 = [(d_1, b_1), \dots, (d_n, b_n \cup b'_n)]$
 - (b) $\mathcal{C}_1 \cup_+ \mathcal{C}_2 = [(d_1, b_1), \dots, (d_{n-1}, b_{n-1})]$ otherwise;
2. for all other cases, $\mathcal{C}_1 \cup_+ \mathcal{C}_2$ is not defined.

Intuitively, if the same formula belongs to all children of a disjunctive node in a tableau, then it can be lifted ‘up’ to the parent, otherwise, \cup_+ merges two subtrees into one subtree where the formula belongs to all children. Recall that the label of a node is the set of all environments from which the node can be derived.

Definition 5 (Union of labels). *The union of two labels L_1 and L_2 , $L_1 \cup_+ L_2 = L_1 \cup L_2 \cup \{e_1 \cup_+ e_2 \mid e_1, e_2 \in L_1 \cup L_2\}$.*

We can now give a sketch of how labels are computed.

Given a justification graph as in Figure 2, we first compute the *justification closure* J for n_\perp , namely the set of justifications that have n_\perp as a consequent, together with the justifications of the antecedents of those justifications, and so on until we reach justifications whose antecedents are axiom nodes. Initially, the labels of all nodes in J other than axiom nodes are empty, and the label of each axiom node in J contains a single environment consisting of the axiom itself.

The justifications in J are processed in order of their ids. For each justification $j : n_{\phi_1}, \dots, n_{\phi_k} \Rightarrow n_\phi \in J$ in turn, if j is deterministic (corresponds to any inference rule apart from the \sqcup -rule), then for every k -tuple of environments from the labels of $n_{\phi_1}, \dots, n_{\phi_k}$ (every way to derive the premises) we take their \cup_{\leq} union (which means, we only combine derivations on the same branch), remove any of the resulting environments which are subsumed (to guarantee minimality), remove nogoods and, if the label of n_ϕ has changed as a result, propagate the changes to the nodes reachable by following already processed (having a smaller id) justification links from n_ϕ (since we discovered a new way to derive those formulas, too).

If a justification $j : n_d \Rightarrow n_{\psi_i}$ is non-deterministic (corresponds to \sqcup -rule), then we need to make sure that the choices corresponding to splitting d are added at the correct points in the tableau tree (recall Figure 1). New branches should be added under each existing branch in the tableau where the disjunction is derivable. To reflect this tableau structure in the label of n_{ψ_i} , for each environment $(\mathcal{A}, \mathcal{C})$ appearing in the label of n_d we compute the set of choice sequences of maximal length appearing in any label, $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$. Each element \mathcal{C}_s ($1 \leq s \leq k$) of this set is maximal (corresponds to a complete branch ending in a leaf), and \mathcal{C} is a prefix of \mathcal{C}_s . For each such \mathcal{C}_s we add an environment $(\mathcal{A}, \mathcal{C}_s + (d, \{\psi_i\}))$ to a set L which is a new set of environments for ψ_i generated by j . For example, in Figure 1, when the \sqcup -rule is applied to

$(C_1 \sqcup C_2)(b)$, the only choice sequence appearing in its label is $[\]$. The set of choice sequences of maximal length which have $[\]$ as a prefix are $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a))]$, $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_2(a))]$, and $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_3(a))]$. The choice sequences in the environments of $C_1(b)$ and $C_2(b)$ become $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a)), ((C_1 \sqcup C_2)(a), C_1(a))]$, $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a)), ((C_1 \sqcup C_2)(a), C_2(a))]$, etc. Finally we add L to the old label of ψ_i using \cup_+ , remove any subsumed environments and no-goods, and propagate the changes to the nodes reachable from ψ_i by following already processed justifications from ψ_i in J .

The label computation algorithms are correct, in that every set of axioms Γ' from which \perp can be derived given the set of justifications produced by the reasoner is a superset of the axioms appearing in some environment in the label of n_{\perp} , and \perp can be derived from every environment in its label.

5 Experimental Results

We have developed a prototype implementation of our approach.² Both the reasoner and the D-ATMS are implemented in Pop-11.³ The tableaux reasoner is implemented as a set of six inference rules using Poprulebase, a Pop-11 rule interpreter.

To evaluate our approach, we performed experiments in which we compared the performance of our prototype system when providing all minimal explanations for inconsistencies in a variety of unfoldable \mathcal{ALC} TBoxes with that of MUPSter [23] and Pellet [24] (version 2.2.2). We chose to compare the D-ATMS with MUPSter and Pellet as they represent different approaches to finding all minimal explanations for an inconsistency. Both use a glass-box approach (extending the reasoner with dependency tracking), but MUPSter finds all minimal explanations, while Pellet finds a single minimal explanation, which is then combined with Reiter's Hitting Set algorithm [19] to find all other explanations [9, 24]. (In our experiments, we used Pellet's glass-box approach, as this typically requires less time to find an explanation [9].) The experiments were performed on a Intel Dual Core 2.16GHz, 2GB RAM PC running Ubuntu. All times are CPU times in ms and represent the average of 5 runs. Only the time actually used for generating explanations is given. We do not count the time AOD, MUPSter, and Pellet spend parsing and loading the ontologies, nor the time required for them to render the explanations.

To test the correctness of our implementation, we compared the results for AOD with those of MUPSter on the set of 1,611 randomly generated unfoldable \mathcal{ALC} TBoxes used by Schlobach to evaluate the performance of MUPSter [23].⁴ For each ontology, we obtained a list of unsatisfiable concept names from RacerPro before finding all minimal explanations for each unsatisfiable concept name.⁵ The explanations generated

² AOD is available at <http://www.agents.cs.nott.ac.uk/research/logics/ontologies>.

³ <http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html>

⁴ The dataset is available at <http://www.few.vu.nl/~schlobac/software.html>.

⁵ <http://www.racer-systems.com/products/racerpro>

by both systems were the same, apart from one case where MUPSter returned a non-minimal explanation.⁶

We also recorded the CPU time required for AOD, MUPSter and Pellet to generate explanations for each ontology. In one case MUPSter did not produce an explanation within 5000 seconds and the run was aborted. We omitted this case and the case in which MUPSter returned a non-minimal explanation from our analysis, and in the following we consider only the remaining 1609 cases. Overall, AOD was noticeably faster than both MUPSter and Pellet, with an average execution time of 30ms (median 9ms) compared to 1001ms (median 166ms) for MUPSter and 478ms (median 383) for Pellet.

To evaluate the performance of AOD on more realistic examples, we used the Geo ontology [23], the Biochemistry-primitive ontology from the TONES repository,⁷ a fragment of the Ordnance Survey BuildingsAndPlaces ontology,⁸ and the Adult Mouse Brain Ontology from the NCBO BioPortal.⁹ The Biochemistry-primitive, BuildingsAndPlaces, and Adult Mouse Brain ontologies were translated into \mathcal{ALC} by removing axioms for inverse roles and role inclusions. As in [23], the Geo ontology was made incoherent by adding disjointness axioms of the form $DJ(A_1, \dots, A_n)$ stating that the concepts A_1, \dots, A_n are pairwise disjoint. To handle the disjointness axioms, we added the following inference rule to the reasoner:

dj-rule from $A_i(a)$ and $DJ(A_1, \dots, A_n)$ derive $\neg A_j(a)$ for all $j \neq i, j \in \{1, \dots, n\}$.

To make the other ontologies incoherent, we choose to systematically create unsatisfiable concepts from existing ontology entailments, allowing us to control the number of unsatisfiable concepts and the form of the resulting explanations. For each ontology, we randomly selected 10 pairs of concepts (A, B) where $A \sqsubseteq B$ is non-trivially entailed by the ontology, i.e., $A \sqsubseteq B \notin \mathcal{T}$. Then for each entailment, $A \sqsubseteq B$, we created a concept $EntailmentA_B \sqsubseteq A \sqcap \neg B$. Finding all minimal explanations for the entailment $A \sqsubseteq B$ thus becomes equivalent to finding all minimal explanations for the unsatisfiability of $EntailmentA_B$.

Table 1: Average execution times for AOD, MUPSter and Pellet.

Ontology	Axioms	Unsat concepts	AOD	MUPSter	Pellet
Geo	500	11	72	259	3649
Biochemistry-primitive	265	10	20	70	418
BuildingsAndPlaces	124	10	42	88	515
Adult Mouse Brain	3447	10	802	1381	3443

⁶ For the TBox `tbox_50_6_1_1_3_5_v1` and unsatisfiable concept `A49` MUPSter returns $\{A49, A37, A26, A34, A0\}$ as an explanation for the unsatisfiability of `A49`, while the D-ATMS returns $\{A49, A37, A34, A0\}$.

⁷ <http://owl.cs.manchester.ac.uk/repository>

⁸ <http://www.ordnancesurvey.co.uk/oswebsite/ontology/BuildingsAndPlaces/v1.1/BuildingsAndPlaces.owl>

⁹ <http://bioportal.bioontology.org/ontologies/1290>

The results are presented in Table 1. The second and third columns show the number of axioms and the total number of unsatisfiable concepts in each ontology. As can be seen, AOD is 1.5 to 3.5 times faster than MUPSter and 4 to 50 times faster than Pellet on these ontologies.

6 Related Work

Two main approaches to axiom pinpointing have been proposed in the literature: glass-box methods and black-box methods. A glass-box method extends a description logic reasoner with some method of dependency tracking. A black-box method, e.g., [9], uses the reasoner as an oracle to determine whether a set of axioms results in an inconsistency or a concept is unsatisfiable with respect to a set of axioms, and then shrinks that set to find a minimal set of reasons for the inconsistency or concept unsatisfiability. Black-box methods have the advantage of being reasoner-independent. However it can be argued that glass-box methods provide additional information in the form of a derivation, which is useful for debugging, e.g., to present the user with a summary of the derivation and which parts of the axioms were used to derive a contradiction.

AOD adopts a glass-box approach to ontology debugging. To date, much of the work on glass box approaches, e.g., [22, 8, 16, 15], has been tailored to a particular logic. More recently, Baader and Peñaloza [3] have proposed a generic tableau rule specification format and a pinpointing algorithm that works for reasoners specified in this format. They also show that termination of a tableau reasoner for satisfiability does not necessarily lead to the termination of its pinpointing extension. In addition, for tableau reasoners that require a blocking condition for termination, e.g., full \mathcal{ALC} , it is not sufficient for the pinpointing extension to use the same blocking condition as the reasoner, because the pinpointing extension needs to take into account not only the presence of an assertion in \mathcal{A} , but also its justifications to determine if a tableau rule instance should be blocked. In [3] they give a characterisation of a class of terminating tableaux where the blocking condition yields a complete and terminating pinpointing extension. However, to the best of our knowledge, this approach has not been implemented. In [18] we sketched an approach to using an ATMS for ontology debugging in a description logic without disjunctions, but did not provide an implementation.

The ATMS as described in [11] does not support non-deterministic choices. However several approaches to handling disjunctions in an ATMS have been proposed in the literature. In [12] de Kleer extended the original ATMS to encode disjunctions of assumptions (axioms) by introducing a set of hyper-resolution rules. However, such rules may significantly reduce the efficiency of the ATMS. Another approach uses a justification for \perp by negated assumptions to represent a disjunction of assumptions, e.g., $A \vee B$ can be encoded by the justification $\neg A, \neg B \Rightarrow \perp$ [13]. Both of these approaches are limited to encoding a disjunction of assumptions (axioms). However, in ontology debugging, disjunctions often appear in concept descriptions. In contrast, the D-ATMS allows disjunctions of nodes corresponding to arbitrary formulas. In [20] the original ATMS was generalised to a clause management system (CMS) where justifications are arbitrary disjunctive clauses. To find the ‘minimal support’ for a clause, the CMS implementation described in [14] uses a method for computing prime implicants which

relies on justifications being clauses consisting of literals to which the resolution rule can be applied. Adopting such an approach would require translating TBox axioms into clauses, and more importantly, finding some way of mapping the clauses returned by the CMS back to the original TBox. The latter in particular is a non-trivial problem. Label computation in the D-ATMS has some similarities with lazy label evaluation in assumption-based truth maintenance systems, e.g., [10], and the restriction to n_{\perp} can be seen as a special case of focussing the ATMS, e.g., [6]. Such approaches have been shown to offer significant performance improvements relative to the ATMS described in [11].

7 Conclusion

We described AOD, a system for debugging unfoldable \mathcal{ALC} TBoxes based on an ATMS with disjunctions. Our approach is correct and complete with respect to a reasoner for \mathcal{ALC} with unfoldable TBoxes. We presented experimental results which suggest that its performance compares favourably with that of MUPSter and Pellet. As the D-ATMS maintains an explicit justification structure, it is straightforward to generate explanations of *how* a contradiction is derivable intended for human users — the D-ATMS essentially keeps intermediate steps in a derivation and can produce them on request.

We believe the D-ATMS is a promising new approach to ontology debugging. Although our approach was developed for \mathcal{ALC} with unfoldable TBoxes, the reasoner and the reason maintenance component are only loosely coupled, and the D-ATMS can be adapted to work with other tableau reasoners. Characterising the conditions under which a terminating tableau algorithm can be combined with the D-ATMS to produce a debugging tool that will find all minimal explanations of \perp is further work. One possible approach would be to build on the results of [3]. The production of more user-friendly explanations of how a contradiction is derivable is also a topic of future work.

References

1. Baader, F., Hollunder, B.: Embedding defaults into terminological representation systems. *Journal of Automated Reasoning* 14, 149–180 (1995)
2. Baader, F., Hollunder, B.: A terminological knowledge representation system with complete inference algorithms. In: *Processing Declarative Knowledge, LNCS*, vol. 567, pp. 67–86 (1991)
3. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. *Journal of Logic and Computation* 20(1), 5–34 (2010)
4. Bail, S., Horridge, M., Parsia, B., Sattler, U.: The justificatory structure of the NCBO BioPortal ontologies. In: *Proceedings of the 10th International Semantic Web Conference (ISWC 2011)*. LNCS, vol. 7031, pp. 67–82. Springer (2011)
5. Broekstra, J., Kampman, A.: Inferencing and truth maintenance in RDF schema. In: *Proceedings of the First International Workshop on Practical and Scalable Semantic Systems*. CEUR Workshop Proceedings, vol. 89. CEUR-WS.org (2003)
6. Forbus, K.D., de Kleer, J.: Focusing the ATMS. In: *Proceedings of the Seventh National Conference on Artificial Intelligence*. pp. 193–198. AAAI Press/MIT Press (1988)

7. Guo, Y., Heflin, J.: An initial investigation into querying an untrustworthy and inconsistent web. In: Proceedings of the ISWC'04 Workshop on Trust, Security, and Reputation on the Semantic Web. vol. 127. CEUR-WS.org (2004)
8. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.: Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics* 3(4), 268–293 (2005)
9. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. *The Semantic Web* pp. 267–280 (2008)
10. Kelleher, G., van der Gaag, L.: The LazyRMS: Avoiding work in the ATMS. *Computational Intelligence* 9(3), 239–253 (1993)
11. de Kleer, J.: An assumption-based TMS. *Artificial Intelligence* 28(2), 127–162 (1986)
12. de Kleer, J.: Extending the ATMS. *Artificial Intelligence* 28(2), 163–196 (1986)
13. de Kleer, J.: A general labeling algorithm for assumption-based truth maintenance. In: Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI'88). pp. 188–192. AAAI Press/MIT Press (1988)
14. de Kleer, J.: An improved incremental algorithm for generating prime implicates. In: Proc. of the Tenth National Conference on Artificial Intelligence (AAAI'92). pp. 780–785. AAAI Press/MIT Press (1992)
15. Lam, J.S.C., Sleeman, D.H., Pan, J.Z., Vasconcelos, W.W.: A fine-grained approach to resolving unsatisfiable ontologies. *Journal of Data Semantics* 10, 62–95 (2008)
16. Meyer, T.A., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic ALC. In: Proceedings of the Twenty First National Conference on Artificial Intelligence (AAAI'06) (2006)
17. Nebel, B.: Terminological reasoning is inherently intractable. *Artificial Intelligence* 43(2), 235–249 (1990)
18. Nguyen, H., Alechina, N., Logan, B.: Ontology debugging with truth maintenance systems. In: Bundy, A., Lehmann, J., Qi, G., Varzinczak, I.J. (eds.) ECAI-10 Workshop on Automated Reasoning about Context and Ontology Evolution (ARCOE-10), Workshop Notes. pp. 13–14. Lisbon, Portugal (2010)
19. Reiter, R.: A theory of diagnosis from first principles. *Artificial Intelligence* 32(1), 57–95 (1987)
20. Reiter, R., de Kleer, J.: Foundations of assumption-based truth maintenance systems: Preliminary report. In: Proceedings of the Sixth National Conference on Artificial Intelligence, (AAAI'87). pp. 183–189 (1987)
21. Ren, Y., Pan, J.Z.: Optimising ontology stream reasoning with truth maintenance system. In: Proceedings of the ACM Conference on Information and Knowledge Management (CIKM 2011) (2011)
22. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03). pp. 355–360. Morgan Kaufmann (2003)
23. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. *Journal of Automated Reasoning* 39(3), 317–349 (2007)
24. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics* 5(2), 51–53 (2007)

Combining DL-Lite with Spatial Calculi for Feasible Geo-thematic Query Answering

Özgür Lütfü Özçep and Ralf Möller

Institute for Software Systems (STS)
Hamburg University of Technology
Hamburg, Germany
{oezguer.oezcep,moeller}@tu-harburg.de

Abstract. First order logic (FOL) rewritability is a desirable feature for query answering over geo-thematic ontologies because in most geo-processing scenarios one has to cope with large data volume. Hence, there is a need for combined geo-thematic logics that provide a sufficiently expressive query language allowing for FOL rewritability. The DL-Lite family of description logics is tailored towards FOL rewritability of query answering for unions of conjunctive queries, hence it is a suitable candidate for the thematic component of a combined geo-thematic logic. We show that a weak coupling of DL-Lite with the expressive region connection calculus RCC8 allows for FOL rewritability under a spatial completeness condition for the ABox. Stronger couplings allowing for FOL rewritability are possible only for spatial calculi as weak as the low-resolution calculus RCC2. Already a strong combination of DL-Lite with the low-resolution calculus RCC3 does not allow for FOL rewritability.

Keywords: FOL rewritability, region connection calculus, qualitative spatial reasoning, GIS, combinations

1 Introduction

Query answering over a database becomes far more difficult if the extensional knowledge in the database is extended by constraints in an ontology. The reason is that a database plus an ontology may have many different models, hence ontology based query answering has to compute the answers w.r.t. to all models and build their intersection (*certain answer* semantics). But in some cases—when using a lightweight logic like DL-Lite for the representation of the ontology and a restricted query language like unions of conjunctive queries—query answering w.r.t. an ontology can be reduced to model checking. This is formalized by the notion of *FOL (first order logic) rewritability*: a given query can be rewritten into a FOL query in which the intensional knowledge of the ontology is captured. Though the rewritten queries may become exponentially bigger than the original queries, there exist optimizations [13]. So, FOL rewritability means a benefit.

DL-Lite per se [2] is not sufficient for use in scenarios of geographic information processing, as these demand among others the representation and deduction over spatial concepts. Though there exists related work combining spatial and thematic reasoning [4], [15], [6], it is not aimed at FOL rewritability. Hence, there is still a need for investigating combinations of logics that, on the one hand, are sufficiently expressive to fit the representation's needs in geographical information processing and that, on the other hand, allow for computationally feasible (in particular FOL rewritable) satisfiability checking and query answering.

Continuing previous work [7, 8], we investigate combinations of logics in the DL-Lite family with different members of the region connection calculus (RCC) family [9], a well-known family of calculi for qualitative spatial reasoning. The DL-Lite logic we are using as thematic part differs from the known members of the DL-Lite family as it also allows for concept conjunctions on the left-hand side of general inclusion axioms. In previous work [8], we focussed on the FOL rewritability aspects for weak combinations of DL-Lite with RCC8; these combinations are weak in so far as they do not allow for the construction of arbitrary RCC8 constraint networks in the intensional part (TBox) of the ontology. In this paper we focus on strong combinations of DL-Lite with the weaker fragments RCC3 and RCC2, and prove that $\text{DL-Lite}_{\mathcal{F},\mathcal{R}}^{\square}(\text{RCC3})$ does not allow for FOL rewritability of satisfiability checking while the weaker $\text{DL-Lite}_{\mathcal{F},\mathcal{R}}^{\square}(\text{RCC2})$ does.

The paper is structured as follows. Section 2 collects technical details on the region connection calculus and the DL-Lite family of description logics. Weak combinations of DL-Lite with the region connection calculus are described in Sect. 3. In Sect. 4, the last section before the conclusion, we consider strong combinations of DL-Lite with weaker fragments of the region connection calculus.

2 Logical Preliminaries

We recapitulate the main logical notation and concepts used in this paper; the region connection calculus and DL-Lite.

2.1 The Region Connection Calculus

We will consider different fragments of the region connection calculus [9] as potential candidates for the spatial logic to be combined with DL-Lite. Randell and colleagues' axiom system [9] is based on a primitive binary relation C intended to denote a connectedness relation which is specified to be reflexive and symmetric. On the basis of C other binary relations between regions which are called *base relations* are explained. One set of base relations is the set \mathcal{B}_{RCC8} , which is the main component of the most expressive region connection calculus RCC8. The base relations of \mathcal{B}_{RCC8} and their intended meanings are given as follows: $\mathcal{B}_{RCC8} = \{\text{DC (disconnected), EC (externally connected), EQ (equal), PO (partially overlapping), NTPP (non-tangential proper part), TPP (tangential proper part), NTPPi (inverse of NTPP), TPPI (inverse of TPP)}\}$. We skip

the concrete definitions of the base relations by the connectedness relation C (see, e.g., [11, p. 45]), as we—in contrast to the axiom system of Randell and colleagues—consider the following axiom system schema Ax_{RCCi} , which directly specifies the properties of the base relations in \mathcal{B}_{RCCi} .

Definition 1 (Axiom system schema Ax_{RCCi}). For all $i \in \{2, 3, 5, 8\}$ the axiom set Ax_{RCCi} contains the following axioms:

$$\begin{aligned} & \{\forall x, y. \bigvee_{r \in \mathcal{B}_{RCCi}} r(x, y)\} \cup && \text{(joint exhaustivity)} \\ & \{\forall x, y. \bigwedge_{r_1, r_2 \in \mathcal{B}_{RCCi}, r_1 \neq r_2} r_1(x, y) \rightarrow \neg r_2(x, y)\} \cup && \text{(pairwise disjointness)} \\ & \{\forall x, y, z. r_1(x, y) \wedge r_2(y, z) \rightarrow r_3^1(x, z) \vee \dots \vee r_3^k(x, z) \mid r_1; r_2 = \{r_3^1, \dots, r_3^k\}\} && \text{(weak composition axioms)} \end{aligned}$$

For $i \in \{3, 5, 8\}$ additionally the axiom $\forall x \text{EQ}(x, x)$ (reflexivity of EQ) is contained. For $i = 2$ one has instead the axiom $\forall x \text{O}(x, x)$ (reflexivity of O).

In particular, the axioms state the JEPD-property of the base relations (each pair of regions x, y is related over exactly one base relation) and describe the (weak) composition of two base relations (denoted by $;$) according to the composition table for RCCi. With the composition of two base relations, in most cases, only indefinite knowledge of spatial configurations follows. The spatial configuration $r_1(x, z) \vee \dots \vee r_n(x, z)$ for base relations r_j in \mathcal{B}_{RCCi} is also written as $\{r_1, \dots, r_n\}(x, z)$, and the set $\{r_1, \dots, r_n\}$ is called a general RCCi relation. Let Rel_{RCCi} be the set of all 2^i general RCCi relations. An RCCi (constraint) network consists of assertions of the form $\{r_1, \dots, r_n\}(x, y)$.

We mention here the composition table for the low resolution logics RCC2 and RCC3. Their base relations are given by the sets $\mathcal{B}_{RCC3} = \{\text{DR}, \text{EQ}, \text{ONE}\}$ and $\mathcal{B}_{RCC2} = \{\text{DR}, \text{O}\}$, and their weak compositions are defined as shown in Fig. 1. The discreteness relation DR is the same as $\{\text{DC}, \text{EC}\}$, the overlapping-but-not-equal relation ONE is equal to $\{\text{PO}, \text{NTPP}, \text{TPP}, \text{NTPPi}, \text{TPPi}\}$ and the overlapping relation O is given by $\{\text{ONE}, \text{EQ}\}$.

$;$	DR	O
DR	\mathcal{B}_{RCC2}	\mathcal{B}_{RCC2}
O	\mathcal{B}_{RCC2}	\mathcal{B}_{RCC2}

$;$	DR	ONE	EQ
DR	\mathcal{B}_{RCC3}	$\{\text{DR}, \text{ONE}\}$	DR
ONE	$\{\text{DR}, \text{ONE}\}$	\mathcal{B}_{RCC3}	ONE
EQ	DR	ONE	EQ

Fig. 1. Composition tables for RCC2 and RCC3

Note that in the definitions of the base relations (of RCC3 and RCC2) we followed the author of [14]. The base relations of the low resolution calculus of [3] are different from those of RCC2; the authors of [3] consider the two base relations DC and $\{\text{EC}, \text{O}\}$. But as we do not deal with the exact definitions of the base relations referring to the connectedness relation C but with the axiom systems referring to the composition table this difference has no effect—the low resolution calculus of [3] has the same trivial composition table as RCC2. For the composition tables of RCC5 and RCC8 have a look at [12, p. 45].

2.2 DL-Lite

The family of DL-Lite description logics [2] is an appropriate candidate for the thematic component of the envisioned geo-thematic logic as it offers computationally feasible satisfiability checking and query answering over ontologies and data stored in a relational database. More concretely, satisfiability checking and query answering (for unions of conjunctive queries) are first order logic (FOL) rewritable. In this paper, we will mainly deal with a member of the extended DL-Lite family which we termed $\text{DL-Lite}_{\mathcal{F},\mathcal{R}}^{\square}$, and which allows functional roles, role hierarchies and role inverses as well as the conjunction of basic concepts on the left-hand side of GCIs (general concept inclusions). The syntax is given in Def. 2. The semantics of this logic is defined in the usual way—but imposing the unique name assumption (UNA).

Definition 2 (DL-Lite $_{\mathcal{F},\mathcal{R}}^{\square}$). Let RN be the set of role symbols and $P \in RN$, CN be a set of concept symbols and $A \in CN$, $Const$ be a set of individual constants and $a, b \in Const$.

$$\begin{aligned} R &\longrightarrow P \mid P^{-} & B &\longrightarrow A \mid \exists R & C_l &\longrightarrow B \mid C_l \sqcap B & C_r &\longrightarrow B \mid \neg B \\ \text{TBox}^* &: & & & C_l &\sqsubseteq C_r, (\text{func } R), R_1 \sqsubseteq R_2 \\ \text{ABox} &: & & & A(a), R(a, b) \end{aligned}$$

*) *Restriction:* If R occurs in a functionality axiom, then R and its inverse do not occur on the right-hand side of a role inclusion axiom $R_1 \sqsubseteq R_2$.

FOL rewritability also holds for the logic $\text{DL-Lite}_{\mathcal{F},\mathcal{R}}^{\square}$ which follows from the corresponding FOL rewritability results for the Datalog extension Datalog^{\pm} [1]. We assume that the reader is familiar with the notion of FOL queries, conjunctive queries (CQ) and unions of conjunctive queries (UCQ), their semantics and the notion of certain answers $\text{cert}(Q, \mathcal{A} \cup \mathcal{T})$ of a query w.r.t. to the union of a TBox and an ABox $\mathcal{T} \cup \mathcal{A}$ [2]. Let $DB(\mathcal{A})$ be the minimal Herbrand model of \mathcal{A} . *Checking the satisfiability of ontologies is FOL rewritable* iff for all TBoxes \mathcal{T} there is a boolean FOL query $Q_{\mathcal{T}}$ s.t. for all ABoxes \mathcal{A} : the ontology $\mathcal{T} \cup \mathcal{A}$ is satisfiable iff $DB(\mathcal{A}) \not\models Q_{\mathcal{T}}$. *Answering queries from a subclass \mathcal{C} of FOL queries w.r.t. to ontologies is FOL rewritable* iff for all TBoxes \mathcal{T} and queries $Q = \psi(\mathbf{x})$ in \mathcal{C} there is a FOL query $Q_{\mathcal{T}}$ such that for all ABoxes \mathcal{A} it is the case that $\text{cert}(Q, \mathcal{T} \cup \mathcal{A}) = Q_{\mathcal{T}}^{DB(\mathcal{A})}$. For DL-Lite, FOL-rewritability can be proved w.r.t. to satisfiability as well as w.r.t. answering UCQs [2, Thm 4.14, Thm 5.15]. A crucial part of the proof for FOL rewritability w.r.t. query answering is the perfect rewriting algorithm [2, Fig. 2]. This algorithm works by using positive inclusion axioms of the form $A_1 \sqsubseteq A_2$ as rewriting rules from right to left so that for example an atom $A_2(x)$ occurring in a CQ would produce a new CQ containing $A_1(x)$ instead of $A_2(x)$.

3 Weak Combinations of DL-Lite with RCC

In this section, we recapitulate the results concerning a weak coupling of DL-Lite with the most expressive region connection calculus fragment RCC8, which we

introduced in [7, 8], and provide an example for its use(fulness). This will give us the opportunity to introduce further concepts that are necessary to understand the following discussions on stronger couplings of DL-Lite with the weaker region connection calculi RCC2 and RCC3.

The combination paradigm follows that of Lutz and Miličić [6] who combine \mathcal{ALC} with the RCC8 and, more generally, with ω -admissible concrete domains [6, Def. 5, p. 7]. The combined logic $\mathcal{ALC}(RCC8)$ of [6] is well behaved in so far as testing concept subsumption is decidable. As we aim at FOL rewritability we have to be even more careful in choosing the right combination method.

We use an axiom set T_ω with corresponding properties of an ω -admissible domain for coupling with DL-Lite because axioms are more appropriate for rewriting investigations. The axiom sets Ax_{RCCi} will instantiate T_ω .

We recapitulate the syntax and the semantics of the constructors of [6] that are used for the coupling of the thematic and the spatial domain. A path U (of length at most 2) is defined as l for a fixed attribute l (“has location”) or as $R \circ l$, the composition of the role symbol R with l . We abbreviate $R \circ l$ with \tilde{R} in this paper. The usual notion of an interpretation \mathcal{I} in our combined logic is slightly modified by using two separate domains $\Delta^{\mathcal{I}}$ and $(\Delta^*)^{\mathcal{I}}$. All symbols of the theory T_ω are interpreted relative to $(\Delta^*)^{\mathcal{I}}$. Let r be an RCC-relation of some RCC-fragment. That is, let be given a set of base relations \mathcal{B}_{RCCi} and $r = \{r_1, \dots, r_n\} \equiv r_1 \vee \dots \vee r_n$ for $r_i \in \mathcal{B}_{RCCi}$. Then $l^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times (\Delta^*)^{\mathcal{I}}$; $r^{\mathcal{I}} = r_1^{\mathcal{I}} \cup \dots \cup r_n^{\mathcal{I}}$; $(R \circ l)^{\mathcal{I}} = \{(d, e^*) \in \Delta^{\mathcal{I}} \times (\Delta^*)^{\mathcal{I}} \mid \text{there is an } e \text{ s.t. } (d, e) \in R^{\mathcal{I}} \text{ and } (e, e^*) \in l^{\mathcal{I}}\}$; $(\exists U_1, U_2.r)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{there exist } e_1^*, e_2^* \text{ s.t. } (d, e_1^*) \in U_1^{\mathcal{I}}, (d, e_2^*) \in U_2^{\mathcal{I}} \text{ and } (e_1^*, e_2^*) \in r^{\mathcal{I}}\}$; $(\forall U_1, U_2.r)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{for all } e_1^*, e_2^* \text{ s.t. } (d, e_1^*) \in U_1^{\mathcal{I}}, (d, e_2^*) \in U_2^{\mathcal{I}} \text{ it holds that } (e_1^*, e_2^*) \in r^{\mathcal{I}}\}$.

Now we can define the following combined geo-thematic logic (where a^*, b^* stand for constants intended to be interpreted by regions):

Definition 3 (DL-Lite $_{\mathcal{F}, \mathcal{R}}^{\square}(\text{RCC8})$). Let $r \in \text{Rel}_{RCC8}$ and $T_\omega = Ax_{RCC8}$.

$$\begin{array}{ll} R \longrightarrow P \mid P^- & U \longrightarrow R \mid \tilde{R} \quad B \longrightarrow A \mid \exists R \mid \exists l \\ C_l \longrightarrow B \mid C_l \sqcap B & C_r \longrightarrow B \mid \neg B \mid \exists U_1, U_2.r \\ TBox^*: & C_l \sqsubseteq C_r, (\text{funct } l), (\text{funct } R), R_1 \sqsubseteq R_2 \\ ABox: & A(a), R(a, b), l(a, a^*), r(a^*, b^*) \end{array}$$

**) Restriction: If $(\text{funct } R) \in \mathcal{T}$, then R and R^- do not occur on the right-hand side of a role inclusion axiom or in a concept of the form $\exists U_1, U_2.r$.*

As satisfiability checking of RCC8 constraint networks is NP-complete, there is only a chance to reach FOL rewritability if we assume within the ABox a constraint network which is consistent and complete, i.e., only base relations are allowed as labels; in this case the ABox is called *spatially complete*. It seems to us that for cadastral maps or maps containing areas of administration one can assume pretty safely (almost) spatial completeness. The coupling with RCC8 is so weak that FOL rewritability of satisfiability follows.

Proposition 1. *Checking the satisfiability of DL-Lite $_{\mathcal{F}, \mathcal{R}}^{\square}(\text{RCC8})$ ontologies that have a spatially complete ABox is FOL rewritable.*

The FOL rewritability holds also for query rewriting w.r.t. to a restricted class of treelike queries (termed GCQ^+). The result is achieved by an adapted perfect rewriting algorithm PerfectRef [2, Fig. 13].

Theorem 1. *Answering GCQ^+ -queries w.r.t. $DL-Lite_{\mathcal{F},\mathcal{R}}^{\square}(RCC8)$ ontologies that have a spatially complete ABox is FOL rewritable.*

$DL-Lite_{\mathcal{F},\mathcal{R}}^{\square}(RCC8)$ as well as the logics introduced below are suited for use in scenarios such as that of an engineering bureau planning additional parks in a city [7]. Assume, the bureau has stored geographical data in some database and declares relevant concepts in the TBox: $Park+Lake \sqsubseteq Park$; $Park\4Playing \sqsubseteq Park$; $Park+Lake \sqsubseteq \exists hasLake \circ l.TPP$; $Park\4Playing \sqsubseteq \exists hasPLAr \circ l.TPP$. The ABox \mathcal{A} is derived virtually by mappings from geographical data in a database. In particular, assume that $\{Park+Lake(a), Park\4Playing(a)\} \subseteq \mathcal{A}$. According to \mathcal{A} , the object a is a park with a lake and a park with a playing area but its spatial extension is not known. Now, the engineering bureau asks for all parks with lakes and playing areas such that the playing area is not contained as island in the lake. This can be formalized by the GCQ^+ query: $Q = Park(x) \wedge \exists hasLake \circ l, hasPLAr \circ l. (\mathcal{B}_{RCC8} \setminus \{NTPP\})(x)$. Using the composition entry TPP ; $TPPi = \{DC, EC, PO, TPP, TPPi, EQ\} \subseteq \mathcal{B}_{RCC8} \setminus \{NTPP\}$, the reformulation algorithm produces a UCQ that contains the following CQ: $Q' = (\exists hasLake \circ l, l.TPP)(x) \wedge (\exists l, hasPLAr \circ l.TPPi)(x)$. Rewriting $\exists l, hasPLAr \circ l.TPPi$ to $\exists hasPLAr \circ l, l.TPP$ in combination with the rewriting rule for $A_1 \sqsubseteq A_2$ we get another CQ $Q'' = Park+Lake(x) \wedge Park\4Playing(x)$. Now, Q'' captures (as desired) the object a .

4 Strong Combinations of DL-Lite with RCC

Another way of reaching FOL rewritability for combinations of DL-Lite with RCC is weakening the expressivity of the spatial component. Hence, one may ask whether a combination with the calculus RCC3 or RCC2 [15], both fragments with weak expressibility, allows for weak FOL rewritability w.r.t. satisfiability checks (and query answering). Their potential use as logics for approximating [5] ontologies in more expressible combined logics like $\mathcal{ALC}(RCC8)$ makes the investigation valuable. The logics $DL-Lite_{\mathcal{F},\mathcal{R}}^{\square,+}(RCC2)$ and $DL-Lite_{\mathcal{F},\mathcal{R}}^{\square,+}(RCC3)$ are defined as follows ('+' indicates the strong combination):

Definition 4 (DL-Lite $_{\mathcal{F},\mathcal{R}}^{\square,+}(RCC2)$ and DL-Lite $_{\mathcal{F},\mathcal{R}}^{\square,+}(RCC3)$). *Let $T_{\omega} = Ax_{RCC2}$ resp. $T_{\omega} = Ax_{RCC3}$ and $r \in \mathcal{B}_{RCC2}$ resp. $r \in \mathcal{B}_{RCC3}$*

$$\begin{array}{ll} R \longrightarrow P \mid P^- & U \longrightarrow l \mid \tilde{R} \quad B \longrightarrow A \mid \exists R \\ C_l \longrightarrow B \mid C_l \sqcap B & C_r \longrightarrow B \mid \neg B \mid \exists U_1, U_2.r \\ TBox^*): & C_l \sqsubseteq C_r, (\text{funct } l, R), R_1 \sqsubseteq R_2 \\ ABox: & A(a), R(a, b), l(a, a^*), r(a^*, b^*) \end{array}$$

**) Restriction: If $(\text{funct } R) \in \mathcal{T}$, then R and R^- do not occur on the right-hand side of a role inclusion axiom.*

For RCC3 the strong combination with $DL-Lite_{\mathcal{F},\mathcal{R}}^{\square}$ leads to non-FOL rewritability. The reason lies in the fact that testing the satisfiability of RCC3 is not in the complexity class AC^0 as shown by the following lemma.

Lemma 1. *Checking satisfiability of RCC3 networks is LOGSPACE hard.*

Proof. As is known, the reachability problem in symmetric (undirected) graphs is logspace complete [10]—where graph reachability asks whether for nodes $s, t \in G$ there is a path between s and t . By reducing this problem to the satisfiability test for RCC3 networks we will have shown that the latter problem is LOGSPACE hard itself. So let be given a (symmetric) graph $G = (V, E)$ and nodes $s, t \in V$. We define the network N in the following way (see Figure 2): Let $V = \{v_1, \dots, v_n\}$ be an enumeration of the nodes of G ; w.l.o.g. let $s = v_1$ and $t = v_n$ and let $\mathcal{B} = \mathcal{B}_{RCC3}$. Nodes of N are given by $V \cup \{a\}$ where $a \notin V$. Labelled edges of N are given by: $s\{\text{DR}\}a$; $t\{\text{ONE}\}a$; $v_i\{\mathcal{B}\}a$ for all $i \neq 1, n$; $v_i\{\text{EQ}\}v_j$ if $E(v_i, v_j)$; $v_i\{\mathcal{B}\}v_j$ if $\neg E(v_i, v_j)$. Now we show that the network N is satisfiable iff s and t are

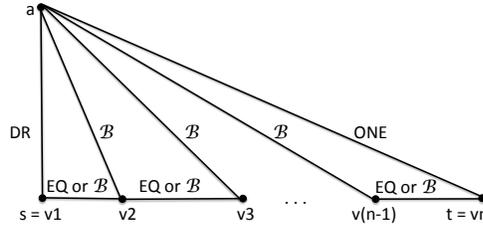


Fig. 2. Network N used in proof of Lemma 1

connected in G . Assume that s and t are connected; then there is an EQ-path in N between them, hence $s\{\text{EQ}\}t$ follows. But this contradicts $s\{\text{DR}\}a$ and $t\{\text{ONE}\}a$. Now assume that s and t are not connected; then there is no path consisting only of EQ-labels between s and t . The graph G consists of at least 2 components, and s, t are in different components. We define a consistent configuration as follows: For all nodes v, v' in the component in which s is contained, let $v\{\text{DR}\}a$ and $v\{\text{EQ}\}v'$. For all nodes v, v' in the component of t let $v\{\text{ONE}\}a$ and $v\{\text{EQ}\}v'$. For all nodes v, v' in the other components let $v\{\text{DR}\}a$ and $v\{\text{EQ}\}v'$. For all nodes v, v' which have not a label yet, let $v\{\text{DR}\}v'$. (Two remarks : 1) EQ-edges for edges $E(v_i, v_j)$ in G with $j > i + 1$ are not shown in Fig. 2. 2) We inserted edges labelled \mathcal{B} for better illustrations. But these are not needed.)

This lemma immediately entails the fact that satisfiability checking for ontologies over the logic $\text{DL-Lite}_{\mathcal{F}, \mathcal{R}}^{\square, +}(\text{RCC3})$ is not FOL rewritable. This problem does not vanish if we presuppose that the ABox \mathcal{A} is spatially complete—as shown by the following proposition.

Proposition 2. *Satisfiability checking of ontologies in $\text{DL-Lite}_{\mathcal{F}, \mathcal{R}}^{\square, +}(\text{RCC3})$ with spatially complete ABoxes is not FOL rewritable.*

Proof. We construct a generic TBox \mathcal{T}_g that allows one to encode any RCC3 constraint network so that checking the consistency of RCC3 constraint networks

is reducible to a satisfiability check of this TBox and a spatially complete ABox. Let for every $r \in Rel_{RCC3}$ be given role symbols R_r^1, R_r^2 . The generic TBox \mathcal{T}_g has for every $r \in Rel_{RCC3}$ a concept symbol A_r and a corresponding axiom with the content that all instances of A_r have paths over the abstract features R_1 resp. R_2 to regions that are r -related.

$$\mathcal{T}_g = \{A_r \sqsubseteq \exists \tilde{R}_r^1, \tilde{R}_r^2 . r, (\text{funct } l, R_r^1, R_r^2) \mid r \in Rel_{RCC3}\} \quad (1)$$

Now, let N be an arbitrary RCC3 constraint network which has to be tested for relational consistency. Let \mathcal{A}_N be an ABox such that for every $r(a, b)$ in N three new constants are introduced: x_{ab}, x_a, x_b .

$$\mathcal{A}_N = \{A_r(x_{ab}), R_r^1(x_{ab}, x_a), R_r^2(x_{ab}, x_b) \mid r(a, b) \in N\} \quad (2)$$

The construction entails: $\mathcal{T}_g \cup \mathcal{A}_N \cup Ax_{RCC3}$ is satisfiable iff $N \cup Ax_{RCC3}$ is satisfiable. If the data complexity of the satisfiability check for DL-Lite $_{\mathcal{F}, \mathcal{R}}^{\square, +}$ (RCC3)-ontologies were in AC^0 , then the consistency of constraint networks could be tested in AC^0 , too. (Note that \mathcal{T}_g is a fixed TBox.) But checking the consistency of RCC3 constraint networks is LOGSPACE-hard and $AC^0 \subsetneq LOGSPACE$.

As a corollary to this proposition we get the assertion that strong combinations of RCC5 and RCC8 into DL-Lite $_{\mathcal{F}, \mathcal{R}}^{\square, +}$ (RCC5) and DL-Lite $_{\mathcal{F}, \mathcal{R}}^{\square, +}$ (RCC8), respectively—which are defined in the same manner as in Definition 4—do not allow for FOL rewritability of satisfiability checking.

The low resolution calculus RCC2 is quite more inexpressive than RCC3 due to the fact that the composition table does not allow for the propagation of information: All compositions of DR, O result in the maximally unspecified relation $\{DR, O\}$. Hence, FOL rewritability of satisfiability testing follows easily considering the query $Q = \exists x, y [O(x, y) \wedge DR(x, y)] \vee \exists x [DR(x, x)]$.

Proposition 3. *Testing the satisfiability of RCC2 networks is FOL rewritable.*

But in combination with functionality axioms of the TBox one could have the problem that the ABox may lead to identifications of regions. The identified regions are not allowed to have edges labelled O, DR resp. to the same region. Though this can be tested, the problem arises when a chain of regions is identified by the TBox and the ABox, because we do not know the length of the chain in advance. More formally: In addition to RCC2 constraint-network assertions we allow identity assertions $v = w$ for regions v, w . As we can assume that all nodes in a RCC2 network are connected by an edge labelled O, DR or \mathcal{B}_{RCC2} we use a more intuitive formalism where, for every assertion $v = w$, the label of the edge between v and w is marked with an =; e.g., an edge between v, w with label $DR^=$ stands for $DR(v, w) \wedge v = w$. We call such a network an =-marked RCC2 network (a RCC $^=$ 2 network for short). Let $\mathcal{B} = \mathcal{B}_{RCC2}$ in the following.

Proposition 4. *An RCC $^=$ 2 constraint network N is unsatisfiable iff*

1. N contains $DR(v, v)$ or $DR^=(v, v)$ for some node v ; or

2. N contains $\text{DR}^-(v, w)$; or
3. N contains a cycle in which there is $\text{DR}(v, w)$ and in which there is a path from v to w such that every label on the path is \mathcal{B}^- or \mathcal{O}^- ; or
4. N contains a cycle in which there is $\text{DR}(v, w)$ and in which there is a path from v to w s.t. every label on the path is \mathcal{B}^- or \mathcal{O}^- except one which is \mathcal{O} .

Proof. Sufficiency of the condition for unsatisfiability is clear. The proof for necessity is done by induction on the number of $=$ -marked labels. So let be given a RCC^- network N . We may assume, that the network has for every pair of nodes exactly one labelled edge between; we assume the edges to be undirected as all relations are symmetric.

Base case: Assume that none of the four conditions hold. As there are no marked labels, then N unsatisfiability can occur only, if N contains $\text{DR}(v, v)$ for some node v (first condition) or $\text{DR}(v, w)$ and $\mathcal{O}(v, w)$ (fourth condition). But these cases are excluded by assumption.

Induction step: Let N contain n marked labelled edges and assume that for all networks N' with $n-1$ marked labelled edges unsatisfiability of N' implies one of the conditions. Now, assume that for N no one of the four conditions holds. We have to show that N is satisfiable. Take an arbitrary $=$ -marked labelled edge between nodes v, w . The label λ of this edge is either \mathcal{O}^- or \mathcal{B}^- . We define a new network N' which results as a contraction from N by identifying v and w to the node z . For N' we may assume again that it contains for every pair of nodes exactly one labelled edge: If in N we have $r_1(v, k)$ and $r_2(w, k)$, then the edge between z and k in N' results as $r_1 \cap r_2$ and is $=$ -marked iff r_1 or r_2 is marked. Clearly $r_1 \cap r_2$ is not empty, as otherwise we would have a contradiction to the fact that N does not fulfill conditions 3 and 4. Clearly N is satisfiable iff N' is satisfiable. Assume to the contrary that N' is unsatisfiable. Hence, one of the four conditions holds for N' : 1) Assume N' contains $\text{DR}(v', v')$ or $\text{DR}^-(v', v')$ for some node v' . If v' is not z , then also $\text{DR}(v', v') \in N$ resp. $\text{DR}^-(v', v') \in N$ —contradicting the fact that N does not fulfill condition 1. Otherwise $v' = z$, but this cannot be the case either, as there are no self-loops $\text{DR}(v, v)$, $\text{DR}^-(v, v)$, $\text{DR}(w, w)$, $\text{DR}^-(w, w)$ in N nor $\text{DR}(v, w)$ or $\text{DR}(v, w)$. 2) Assume N' contains $\text{DR}^-(v', w')$. If neither v' nor w' is z , this contradicts the fact that N does not fulfill the 2. condition. Otherwise $v' = w' = z$, leading to a contradiction with the fact that N does not fulfill condition 1. 3) Assume N' contains a cycle in which there is $\text{DR}(v', w')$ and there is a path from v' to w' such that every label on the path is \mathcal{B}^- or \mathcal{O}^- . The case that the path does not contain z immediately leads to a contradiction. Otherwise, the path extends to a path in N fulfilling the 3. condition—contradiction. Similarly, if N' fulfills condition 4, the verifying path can be extended to a path in N fulfilling condition 4—contradiction.

Proposition 4 shows that adding identity assertions to an RCC2 network may require checking the existence of identity chains of arbitrary length. Hence, in principle it is possible that the functional roles used in $\text{DL-Lite}_{\mathcal{F}, \mathcal{R}}^{\square, \dagger}(\text{RCC2})$ may lead to identity chains. But as we want to show in the following proposition, this cannot be the case: The identity paths induced by functionalities in $\text{DL-Lite}_{\mathcal{F}, \mathcal{R}}^{\square, \dagger}(\text{RCC2})$ can have only a maximal length of one.

Proposition 5. *Satisfiability checking of ontologies in $DL\text{-Lite}_{\mathcal{F},\mathcal{R}}^{\square,+}(RCC2)$ is FOL rewritable.*

We give a proof sketch. The idea is to find concepts that are unsatisfiable according to the TBox (this amounts to constructing the negative closure as in the proofs for pure DL-Lite [2, Def. 4.7, p. 292]); these are formulated as boolean queries, and the (finite) disjunctions of these queries is answered over the ABox; e.g., if $A_1 \sqsubseteq \neg A_2$ is in the TBox, then the query would contain $\exists x(A_1(x) \wedge A_2(x))$ as disjunct. The introduction of concepts of the form $\exists U_1, U_2.r$ enlarges the potential conflicts of a TBox \mathcal{T} with an ABox \mathcal{A} . So in addition to the FOL queries that result from the negative closure of the TBox, we have to find queries for the potential conflicts in the four conditions of Proposition 4. The resulting conjunctive queries have further to be fed into a perfect rewriting algorithm like PerfectRef for pure DL-Lite [2] in order to capture the implications of the TBox.

Concerning the first two conditions we have to deal with axioms of the form $B \sqsubseteq \exists l, l.DR$. If this axiom occurs in the TBox \mathcal{T} , then one has to produce the CQ $\exists x.B(x)$. Similarly, if $\{B \sqsubseteq \exists \tilde{R}, \tilde{R}.DR, (\text{funct } R)\} \subseteq \mathcal{T}$, then the CQ $\exists x.B(x)$ has to be added. Also if $\{B \sqsubseteq \exists \tilde{R}_1, \tilde{R}_2.DR, (\text{funct } R_1, \text{funct } R_2)\} \subseteq \mathcal{T}$, then we may get a conflict of the first kind and hence we have to add a CQ $\exists x, y[B(x) \wedge R_1(x, y) \wedge R_2(x, y)]$. Concerning the third and fourth condition we note that this can be only the case if the $=$ -marked paths have maximal length one. Otherwise we already have a contradiction of the ABox with the functionality assertions in the TBox. Hence, one considers only the case of pairs of TBox axioms of the general form $A \sqsubseteq \exists \tilde{R}_1, \tilde{R}_2.DR$ and $B \sqsubseteq \exists \tilde{R}_3, \tilde{R}_4.O$ with $(\text{funct } R_1, R_2, R_3, R_4)$. In this case we feed also into the PerfectRef algorithm the CQ $\exists x, y, z, w[A(x) \wedge B(y) \wedge R_1(x, z) \wedge R_3(y, z) \wedge R_2(x, w) \wedge R_4(y, w)]$.

5 Conclusion

As recapitulated in this paper, combining DL-Lite with expressive fragments of the region calculus like RCC8 into logics that preserve the property of FOL rewritability is possible if the coupling is weak: Constraints of the RCC8 network contained in the ABox are not transported over to the implicitly constructed constraint network resulting from the constructors of the form $\exists U_1, U_2.r$. In this paper we mainly dealt with strong combinations for weaker calculi like RCC2 or RCC3. As we have shown by a reduction proof, a strong combination with RCC3 destroys the FOL rewritability of satisfiability checking. The reason is that checking the satisfiability of RCC3 networks needs to test for reachability along EQ paths, which can be reproduced by the TBox. For the low resolution calculus RCC2, FOL rewritability of satisfiability checking is provable—though checking the satisfiability of RCC2 networks with additional identity assertions is at least as hard as checking RCC3 networks. We plan to investigate whether $DL\text{-Lite}_{\mathcal{F},\mathcal{R}}^{\square,+}(RCC2)$ and $DL\text{-Lite}_{\mathcal{F},\mathcal{R}}^{\square}(RCC8)$ can be used for approximation—following the complete but not necessarily correct approximation method of [5]. Moreover we want to check whether $DL\text{-Lite}_{\mathcal{F},\mathcal{R}}^{\square,+}(RCC2)$ allows for FOL rewritability of query answering w.r.t. unions of conjunctive queries.

References

1. Cali, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. Technical Report CL-RR-10-21, Oxford University Computing Laboratory (November 2010)
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Rosati, R.: Ontologies and databases: The DL-Lite approach. In: Tessaris, S., Franconi, E. (eds.) Semantic Technologies for Informations Systems – 5th Int. Reasoning Web Summer School (RW-2009), LNCS, vol. 5689, pp. 255–356. Springer (2009)
3. Grigni, M., Papadias, D., Papadimitriou, C.H.: Topological inference. In: IJCAI (1). pp. 901–907 (1995)
4. Haarslev, V., Lutz, C., Möller, R.: A description logic with concrete domains and a role-forming predicate operator. *J. Log. Comput.* 9(3), 351–384 (1999)
5. Kaplunova, A., Moeller, R., Wandelt, S., Wessel, M.: Towards scalable instance retrieval over ontologies. In: Yaxin, B., Mary-Anne, W. (eds.) Knowledge Science, Engineering and Management, Fourth International Conference, KSEM 2010, Proceedings. Lecture Notes in Computer Science, vol. 6291. Springer (2010)
6. Lutz, C., Miličić, M.: A tableau algorithm for description logics with concrete domains and general TBoxes. *J. Autom. Reasoning* 38(1-3), 227–259 (2007)
7. Özçep, O.L., Möller, R.: Computationally feasible query answering over spatio-thematic ontologies. In: Proceedings of GEOProcessing 2012, The Fourth International Conference on Advanced Geographic Information Systems, Applications, and Services (2012), available online at http://www.thinkmind.org/index.php?view=article&articleid=geoprocessing_2012_7_10_30059
8. Özçep, Ö.L., Möller, R.: Scalable geo-thematic query answering. Technical report, Institute for Softwaresystems (STS), Hamburg University of Technology (2012), available online at <http://www.sts.tu-harburg.de/tech-reports/papers.html>
9. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning. pp. 165–176 (1992)
10. Reingold, O.: Undirected connectivity in log-space. *J. ACM* 55(4), 17:1–17:24 (Sep 2008), <http://doi.acm.org/10.1145/1391289.1391291>
11. Renz, J.: Qualitative Spatial Reasoning with Topological Information, Lecture Notes in Computer Science, vol. 2293. Springer (2002)
12. Renz, J., Nebel, B.: Qualitative spatial reasoning using constraint calculi. In: Aiello, M., Pratt-Hartmann, I., Benthem, J. (eds.) Handbook of Spatial Logics, pp. 161–215. Springer Netherlands (2007)
13. Rodríguez-Muro, M., Calvanese, D.: Semantic index: Scalable query answering without forward chaining or exponential rewritings. In: Posters of the 10th Int. Semantic Web Conf. (ISWC 2011) (2011)
14. Wessel, M.: On spatial reasoning with description logics - position paper. In: Ian Horrocks, S.T. (ed.) Proceedings of the International Workshop on Description Logics (DL-2002). CEUR Workshop Proceedings, vol. 53 (2002), <http://CEUR-WS.org/Vol-53/>
15. Wessel, M.: Qualitative spatial reasoning with the $\mathcal{ALCCIRCC}$ - family — first results and unanswered questions. Technical Report FBI-HH-M-324/03, University of Hamburg, Department for Informatics (2003)

OCL-Lite: A Decidable (Yet Expressive) Fragment of OCL^{*}

Anna Queralt², Alessandro Artale¹, Diego Calvanese¹, and Ernest Teniente²

¹ KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy
{artale,calvanese}@inf.unibz.it

² Dept. of Service and Information System Engineering
UPC – BarcelonaTech, Spain
{aqueralt,teniente}@essi.upc.edu

Abstract. UML has become a de facto standard in conceptual modeling. Class diagrams in UML allow one to model the data in the domain of interest by specifying a set of graphical constraints. However, in most cases one needs to provide the class diagram with additional semantics to completely specify the domain, and this is where OCL comes into play. While reasoning over class diagrams is decidable and has been investigated intensively, it is well known that checking the correctness of OCL constraints is undecidable. Thus, we introduce OCL-Lite, a fragment of the full OCL language and prove that reasoning over UML class diagrams with OCL-Lite constraints is in EXPTIME by an encoding in the description logic \mathcal{ALCT} . As a side result, DL techniques and tools can be used to reason on UML class diagrams annotated with arbitrary OCL-Lite constraints.

1 Introduction

The Unified Modeling Language (UML) has become a de facto standard in conceptual modeling of information systems. In UML, a conceptual schema is represented by means of a class diagram, with its graphical constraints, together with a set of user-defined constraints, which are usually specified in the Object Constraint Language (OCL). In every application domain the set of instances that can be stored or managed is necessarily finite and, thus, a schema has not only to be consistent, but also finitely satisfiable [13]. It is well-known that reasoning with OCL integrity constraints in their full generality is undecidable since it amounts to full FOL reasoning [4]. Thus, reasoning with UML conceptual schemas in the presence of OCL constraints has been approached in the following alternative ways:

* This paper is a shortened version of [14]. This work has been partly supported by the Ministerio de Ciencia e Innovación under the projects TIN2008-03863 and TIN2008-00444, Grupo Consolidado.

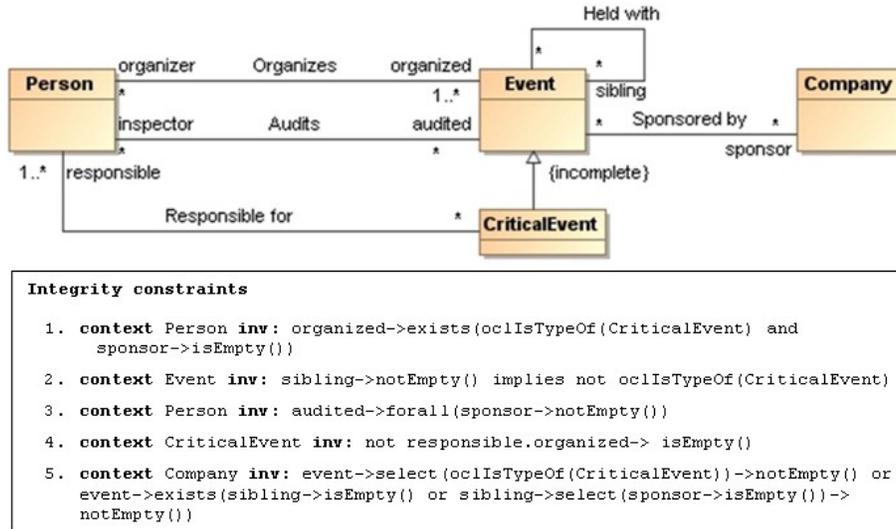


Fig. 1. UML class diagram with OCL constraints

1. Allowing general constraints (in OCL or other languages) without guaranteeing termination in all cases [9, 6, 15].
2. Allowing general constraints and ensuring termination without guaranteeing completeness of the result [1, 17, 12].
3. Ensuring both termination and completeness of finite reasoning by allowing only specific kinds of constraints [13, 11, 18].
4. Ensuring terminating and complete reasoning by disallowing OCL constraints and admitting unrestricted models [8, 5, 10, 3, 2].

To our knowledge, none of the existing approaches guarantees complete and terminating reasoning on UML schemas coupled with OCL constraints able to capture those in Figure 1. With approaches of the first kind, a result may not be obtained in some particular cases. On the contrary, the second kind of approaches may fail to find existing valid solutions. Approaches of the third kind do not allow arbitrary constraints as the ones in Figure 1. Finally, the last approaches are based on a Description Logic (DL) encoding of a UML schema. These methods do not consider OCL constraints and they usually check unrestricted satisfiability.

The main purpose of this paper is to identify a fragment of OCL, which we call OCL-Lite, that guarantees *finite satisfiability* while being significantly expressive at the same time. In other words, we propose the specification of arbitrary constraints within the bounds of OCL-Lite in a UML conceptual schema to ensure completeness and decidability of reasoning. Such nice properties are due to the *finite model property* (FMP) of the language, i.e., it is guaranteed that a satisfiable UML/OCL-Lite schema is always finitely satisfiable. The proposed OCL-Lite is the result of identifying a fragment of OCL that can be encoded into the DL *ALCI* [7], which has interesting reasoning properties. In particu-

lar, \mathcal{ALCI} enjoys the FMP, i.e., every satisfiable set of constraints formalized in \mathcal{ALCI} admits a finite model. Thus, the FMP is also guaranteed for any fragment of OCL that fits into \mathcal{ALCI} .

The contributions of this paper can be summarized as follows:

- The identification of a fragment of OCL that enjoys the FMP. To our knowledge, the reasoning properties of OCL had not been studied before, except that full OCL leads to undecidability.
- A mapping from OCL-Lite to the DL \mathcal{ALCI} to prove that the proposed fragment has the same reasoning properties as \mathcal{ALCI} . To our knowledge, this is the first attempt to encode OCL constraints in DLs. As a side result, a DL reasoner can be used to verify the correctness of a schema.
- Thanks to the mapping to \mathcal{ALCI} we are able to show both the FMP and that checking satisfiability in UML/OCL-Lite is an EXPTIME-complete problem.

2 OCL-Lite Syntax and Semantics

In this section we present the fragment of OCL that corresponds to OCL-Lite. We start by an overview of basic concepts of UML and OCL.

A UML class diagram represents the static view of the domain basically by means of classes and associations between them, representing, respectively, sets of objects and relations between objects. An association is constrained by a set of participating classes. An *association end* is a part of an association that defines the participation of a class in the association. The name of the *role* played by a participant in an association is placed in the association end near the corresponding class. If the role name is omitted, the role played by the participant is the name of its class.

An *OCL constraint* (or invariant) has the form: `context C inv: $OCLExpr$` , where C is the *contextual class*, i.e., the class to which the constraint belongs, and $OCLExpr$ is an expression that results in a boolean value. An OCL constraint is satisfied by an instantiation of the schema if $OCLExpr$ evaluates to true for each instance of the contextual class. An *OCL expression* is defined by means of navigation paths, combined with predefined OCL operations to specify conditions on those paths. A *navigation path* is a sequence of role names defined in the associations of the class diagram. Each role name used in a path indicates the direction of the navigation.

2.1 OCL-Lite Syntax

In this section we specify the syntax rules that allow one to construct OCL constraints belonging to the fragment of OCL that we call OCL-Lite. An *OCL-Lite constraint* has the form: `context C inv: $OCL-LiteExpr$` . In the syntax rules, shown in Figure 2, an *OCL-Lite expression* $OCL-LiteExpr$ is defined recursively. Intuitively, $OCL-LiteExpr$ allows one to construct a boolean OCL-Lite expression, which can correspond to the whole constraint, or can be the condition

```

OCL-LiteExpr ::= Path SelectExpr | oclIsTypeOf(C) | not OCL-LiteExpr |
                OCL-LiteExpr and OCL-LiteExpr |
                OCL-LiteExpr or OCL-LiteExpr |
                OCL-LiteExpr implies OCL-LiteExpr
Path          ::= PathItem | PathItem.Path
PathItem     ::= ri | oclAsType(C).ri
SelectExpr   ::= BooleanOp | ->select(OCL-LiteExpr) SelectExpr
BooleanOp    ::= ->exists(OCL-LiteExpr) | ->forall(OCL-LiteExpr) |
                ->size()>0 | ->size()=0 | ->isEmpty() | ->notEmpty()

```

Fig. 2. Syntax of OCL-Lite expressions

specified as a parameter in a `select` operation (see *SelectExpr*) or in `exists` and `forall` operations (see *BooleanOp*). An *OCL-LiteExpr* can be either a *Path* to which a *SelectExpr* is applied, a check of whether an object is of a certain type, or boolean combinations of these OCL-Lite expressions.

The label *Path* indicates how a navigation path can be constructed as a non-empty sequence of *PathItems*. Each *PathItem* can be either a role name r_i specified in the class diagram, or a role name preceded by the operation *oclAsType(C)*, when we need to access a role name of a particular class C . When *OCL-LiteExpr* corresponds to the whole constraint, each path starts from a role that is accessible from the contextual class (or a subclass C of the contextual class, in which case *oclAsType(C)* must be specified first). Otherwise, when *OCL-LiteExpr* is inside a `select`, `exists`, or `forall` operation, then, the starting role name will depend on the context where the operation is used. After a *Path*, one can apply a (possibly empty) sequence of selections on the collection of objects obtained through the path, always followed by a *BooleanOp*.

The intuitive meaning of the OCL collection operations included in this fragment of OCL is as follows. Let *col* denote the collection of objects reachable along a path, and let *o* denote a single object, then:

- *col->select(OCL-LiteExpr)*: returns the subset of elements of *col* that satisfy *OCL-LiteExpr*;
- *col->exists(OCL-LiteExpr)*: returns true iff there is some element of *col* that satisfies *OCL-LiteExpr*;
- *col->forall(OCL-LiteExpr)*: returns true iff all the elements of *col* satisfy *OCL-LiteExpr*;
- *col->size()*: returns the number of elements of *col*;
- *col->isEmpty()*: returns true iff *col* is empty;
- *col->notEmpty()*: returns true iff *col* is not empty;
- *o.oclIsTypeOf(C)*: returns true iff *o* is an instance of the class C ;

All constraints in Figure 1 are examples of well-formed OCL-Lite expressions.

2.2 OCL-Lite Semantics

OCL-Lite operations, except for *oclIsTypeOf* and *oclAsType*, can be expressed only in terms of `select`, `isEmpty`, and `notEmpty`. Thus, to specify the seman-

Table 1. Normalization of OCL-Lite expressions

	Original expression	Normalized expression
a)	$col \rightarrow \text{exists}(\text{cond})$	$col \rightarrow \text{select}(\text{cond}) \rightarrow \text{notEmpty}()$
b)	$col \rightarrow \text{forall}(\text{cond})$	$col \rightarrow \text{select}(\text{not cond}) \rightarrow \text{isEmpty}()$
c)	$col \rightarrow \text{select}(\text{cond}_1) \rightarrow \text{select}(\text{cond}_2)$	$col \rightarrow \text{select}(\text{cond}_1 \text{ and } \text{cond}_2)$
d)	$col \rightarrow \text{size}() > 0$	$col \rightarrow \text{notEmpty}()$
e)	$col \rightarrow \text{size}() = 0$	$col \rightarrow \text{isEmpty}()$
f)	$\text{not } col \rightarrow \text{isEmpty}()$	$col \rightarrow \text{notEmpty}()$
g)	$\text{not } col \rightarrow \text{notEmpty}()$	$col \rightarrow \text{isEmpty}()$

tics of OCL-Lite expressions, we first rewrite each expression as an equivalent *normalized* one, which is expressed in terms of these operations only. Table 1 shows the OCL-Lite expressions together with their normal form. These normalizations, together with de Morgan’s laws, are iteratively applied until the expression only contains the operations `select`, `isEmpty`, and `notEmpty`, and the boolean operator `not` only appears before `oclIsTypeOf(C)`. In the table, *col* and *cond* denote, respectively, a collection and a condition, which must be defined according to the syntax rules.

In our running example, the constraints in Figure 1 that have to be normalized are 1, 3, 4, and 5. The resulting expressions we get after applying the rules in Table 1 are:

- Constraint 1. We apply rule a) and we get the normalized expression:

```
context Person inv:
    organized->select(oclIsTypeOf(CriticalEvent) and
                    sponsor->isEmpty())->notEmpty()
```
- Constraint 3. We first apply rule b) and we get:

```
context Person inv:
    audited->select(not sponsor->notEmpty())->isEmpty()
```

 We then apply rule g) to obtain the normalized expression:

```
context Person inv:
    audited->select(sponsor->isEmpty())->isEmpty()
```
- Constraint 4. We apply rule f) and we get:

```
context CriticalEvent inv: responsible.organized->notEmpty()
```
- Constraint 5. We apply rule a) and we get:

```
context Sponsor inv:
    event->select(oclIsTypeOf(CriticalEvent))->notEmpty() or
    event->select(sibling->isEmpty() or
                sibling->select(sponsor->isEmpty())->notEmpty())->notEmpty()
```

It can be seen from Table 1 that the expressions resulting from the normalization conform to a limited set of patterns, basically consisting of an optional `select` operation followed either by `isEmpty` or `notEmpty`. Also, the expression may include the operation `oclIsTypeOf`.

In the following we consider OCL-Lite expressions in their normal form. For each of them we specify its semantics by means of an interpretation function, *f*,

that maps each *OCL-LiteExpr* into a FOL formula $OCL-LiteExpr^f(x)$ with one free variable. Other approaches specify the semantics of OCL expressions using first-order terms instead of formulas [4]. However, as also argued in [4], using formulas is preferable when dealing with sets, as in our case.

We start by formalizing the semantics of an OCL-Lite constraint

context C inv: *OCL-LiteExpr*

Its interpretation is: $\forall x (C(x) \rightarrow OCL-LiteExpr^f(x))$ where C is the unary predicate corresponding to class C .

To define the semantics of OCL-Lite expressions, we first introduce some notation to deal with navigation paths. Consider a navigation path $p_n \dots p_1$ in an OCL-Lite expression, where each p_i is either a role name or $oclAsType(C_i) \cdot r_i$. To formalize a (binary) association A_i , we introduce a binary predicate, A_i , whose first argument represents an instance of $dom(A_i)$ (the domain of A_i) and whose second argument represents an instance of $range(A_i)$ (the range of A_i). To fix a semantics for role names we conform to the UML convention about role names [16], i.e., a role name attached to an association end labeled with a class C is used to navigate from one object to another one belonging to the class C . Thus, in the following, $p_i^f(x, y) = A_i(x, y)$ when p_i is a role name attached to the $range(A_i)$ -end of A_i , and, viceversa, $p_i^f(x, y) = A_i(y, x)$ when p_i is a role name attached to the $dom(A_i)$ -end of A_i . Similarly, $p_i^f(x, y) = C_i(x) \wedge A_i(x, y)$, when $p_i = oclAsType(C_i) \cdot r_i$ and r_i is a role name attached to the $range(A_i)$ -end of A_i , while $p_i^f(x, y) = C_i(x) \wedge A_i(y, x)$, when $p_i = oclAsType(C_i) \cdot r_i$ and r_i is a role name attached to the $dom(A_i)$ -end of A_i .

1. $OCL-LiteExpr = p_n \dots p_1 \rightarrow select(OCL-LiteExpr_0) \rightarrow notEmpty()$
The semantics of this expression is

$$OCL-LiteExpr^f(x) = \exists x_n \dots \exists x_1 (p_n^f(x, x_n) \wedge \dots \wedge p_1^f(x_2, x_1) \wedge OCL-LiteExpr_0^f(x_1))$$

A particular case of this expression is when no **select** operation is applied on the objects obtained from the navigation, which corresponds to the expression $OCL-LiteExpr = p_n \dots p_1 \rightarrow notEmpty()$. In this case, we have

$$OCL-LiteExpr^f(x) = \exists x_n \dots \exists x_1 (p_n^f(x, x_n) \wedge \dots \wedge p_1^f(x_2, x_1))$$

2. $OCL-LiteExpr = p_n \dots p_1 \rightarrow select(OCL-LiteExpr_0) \rightarrow isEmpty()$
The semantics of this expression is

$$OCL-LiteExpr^f(x) = \forall x_n \dots \forall x_1 (\neg p_n^f(x, x_n) \vee \dots \vee \neg p_1^f(x_2, x_1) \vee \neg OCL-LiteExpr_0^f(x_1))$$

Again, we have a particular case of this kind of expression in the absence of **select**, namely $OCL-LiteExpr = p_n \dots p_1 \rightarrow isEmpty()$. In this case, the semantics of the expression is

$$OCL-LiteExpr^f(x) = \forall x_n \dots \forall x_1 (\neg p_n^f(x, x_n) \vee \dots \vee \neg p_1^f(x_2, x_1))$$

3. $OCL-LiteExpr = [not] oclIsTypeOf(C)$
where the brackets denote optionality. The semantics of the expression is

$$OCL-LiteExpr^f(x) = [\neg]C(x)$$

4. $OCL\text{-}LiteExpr = OCL\text{-}LiteExpr_1$ and $OCL\text{-}LiteExpr_2$
The semantics of this expression is

$$OCL\text{-}LiteExpr^f(x) = OCL\text{-}LiteExpr_1^f(x) \wedge OCL\text{-}LiteExpr_2^f(x)$$

5. $OCL\text{-}LiteExpr = OCL\text{-}LiteExpr_1$ or $OCL\text{-}LiteExpr_2$
The semantics of this expression is

$$OCL\text{-}LiteExpr^f(x) = OCL\text{-}LiteExpr_1^f(x) \vee OCL\text{-}LiteExpr_2^f(x)$$

6. $OCL\text{-}LiteExpr = OCL\text{-}LiteExpr_1$ implies $OCL\text{-}LiteExpr_2$
The semantics of this expression is

$$OCL\text{-}LiteExpr^f(x) = OCL\text{-}LiteExpr_1^f(x) \rightarrow OCL\text{-}LiteExpr_2^f(x)$$

Definition 1 (Satisfiability of OCL-Lite constraints). Let Γ be a set of OCL-Lite constraints and Γ^f the resulting FOL theory. Then, Γ is said to be satisfiable if there exists a first order interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that satisfies Γ^f . \mathcal{I} is called a model of Γ .

3 Encoding UML/OCL-Lite in \mathcal{ALCI}

In this section we show that the proposed fragments of OCL and UML can both be encoded in the DL \mathcal{ALCI} . Thus, finite reasoning is guaranteed for every conceptual schema expressed in this language. We first present the fragment of UML we are interested in, and then we provide an encoding for the OCL-Lite fragment, too.

We assume the encoding of a fragment of UML class diagrams in \mathcal{ALCI} , based on the encoding in \mathcal{ALCQI} proposed in [5]. Since \mathcal{ALCQI} is an extension of \mathcal{ALCI} that does not have the FMP [7] (i.e., a schema specified in \mathcal{ALCQI} might be satisfiable only by an infinite number of instances), we focus on a fragment of UML that can be encoded into \mathcal{ALCI} . In particular, we consider UML class diagrams where the domain of interest is represented through *classes* (representing sets of objects), possibly equipped with *attributes* and *associations* (representing relations among objects), and *types* (representing the domains of attributes, i.e., integer, string, etc.). The kind of UML constraints that we consider in this paper are the ones typically used in conceptual modeling, namely *hierarchical* relations between classes, *disjointness* and *covering* between classes, *cardinality* constraints for participation of entities in relationships, and *multiplicity* and *typing* constraints for attributes. To preserve the FMP we restrict both cardinality and multiplicity constraints to be of the form $*$ or $1..*$ (meaning that either no constraint applies or the class participates at-least once, respectively).

The encoding, starting from a UML class diagram Σ , generates a satisfiability preserving \mathcal{ALCI} knowledge base \mathcal{K}_Σ (see [5] for details on the encoding). Given the UML fragment chosen, a model of the knowledge base \mathcal{K}_Σ can be viewed as an instantiation of the UML class diagram Σ .

In the following, we provide a mapping to translate OCL-Lite constraints into \mathcal{ALCI} . An OCL-Lite constraint, which has the general form

context C inv: $OCL-LiteExpr$

is encoded as the following \mathcal{ALCI} inclusion assertion:

$$C \sqsubseteq OCL-LiteExpr^\dagger$$

where \cdot^\dagger is a mapping function that assigns to each OCL-Lite expression its corresponding \mathcal{ALCI} encoding. This inclusion assertion, according to the semantics of OCL constraints, states that each instance of C must satisfy $OCL-LiteExpr$. We now illustrate the encoding of OCL-Lite expressions in \mathcal{ALCI} . We consider OCL-Lite expressions in their normal form, as provided in Section 2.2, and define $OCL-LiteExpr^\dagger$ by induction on the structure of $OCL-LiteExpr$.

1. $OCL-LiteExpr = p_n \dots p_1 \rightarrow \text{select}(OCL-LiteExpr_0) \rightarrow \text{notEmpty}()$

We define the \mathcal{ALCI} concept $OCL-LiteExpr^\dagger$ by induction on the length n of the navigation path. For convenience, we consider as base case $n = 0$, and in this case we set $OCL-LiteExpr^\dagger = OCL-LiteExpr_0^\dagger$.

For the inductive case, let $OCL-LiteExpr_n = p_n \dots p_1 \rightarrow \text{select}(OCL-LiteExpr_0) \rightarrow \text{notEmpty}()$, let p_{n+1} be an additional path item, and let $OCL-LiteExpr_{n+1} = p_{n+1} \cdot OCL-LiteExpr_n$. Then $OCL-LiteExpr_{n+1}^\dagger = p_{n+1}^\dagger \cdot (OCL-LiteExpr_n^\dagger)$, where p_{n+1}^\dagger (for the various cases of p_{n+1} , cf. the OCL syntax in Section 2.1) is an abbreviation³ defined as follows (r denotes a role name of an association A , and $dom(A)$ and $range(A)$ denote respectively the domain and range of A):

$$r^\dagger = \begin{cases} \exists A & \text{if } r \text{ is attached to } range(A) \\ \exists A^- & \text{if } r \text{ is attached to } dom(A) \end{cases}$$

$$(\text{oclAsType}(C).r)^\dagger = \begin{cases} C \sqcap \exists A & \text{if } r \text{ is attached to } range(A) \\ C \sqcap \exists A^- & \text{if } r \text{ is attached to } dom(A) \end{cases}$$

Note that the \mathcal{ALCI} concept corresponding to $OCL-LiteExpr$ has the form

$$p_n^\dagger \cdot (p_{n-1}^\dagger \cdot (\dots (p_1^\dagger \cdot (OCL-LiteExpr_0^\dagger)) \dots))$$

Intuitively, this concept represents the fact that $OCL-LiteExpr$ evaluates to true, for a given instance o in the domain of p_n , if o is related through the path $p_n \dots p_1$ to some object o_1 that satisfies the condition specified by $OCL-LiteExpr_0$. When there is no **select** operation, i.e., the OCL expression has the form $p_n \dots p_1 \rightarrow \text{notEmpty}()$, then $OCL-LiteExpr_0^\dagger = \top$, and the constraint is encoded as $p_n^\dagger \cdot (p_{n-1}^\dagger \cdot (\dots (p_1^\dagger \cdot \top) \dots))$.

For example, once it has been normalized in Section 2.2, an expression that follows this pattern is the one in the body of constraint 4. The \mathcal{ALCI} assertion that encodes this constraint is:

$$\text{CriticalEvent} \sqsubseteq \exists \text{ResponsibleFor}^- \cdot (\exists \text{Organizes} \cdot \top)$$

³ Note that p^\dagger is not a valid DL expression.

Note that, the first DL role, ResponsibleFor^- , is inverted since the association ResponsibleFor has domain Person and range CriticalEvent , and the role name responsible is attached to Person . Thus, $\text{responsible}^\dagger = \exists\text{ResponsibleFor}^-$. The next role name in the OCL-Lite expression is organized , which is attached to Event , the range of the association Organizes . Thus, $\text{organized}^\dagger = \exists\text{Organizes}$. Finally, since the OCL operation select does not appear in the constraint, no condition must be imposed on the instances reached at the end of the path.

2. $\text{OCL-LiteExpr} = p_n \dots p_1 \text{->select}(\text{OCL-LiteExpr}_0) \text{->isEmpty}()$

Similarly to the previous case, we define $\text{OCL-LiteExpr}^\dagger$ by induction on n . For the base case of $n = 0$, we set $\text{OCL-LiteExpr}^\dagger = \neg\text{OCL-LiteExpr}_0^\dagger$. For the inductive case, let:

$$\text{OCL-LiteExpr}_n = p_n \dots p_1 \text{->select}(\text{OCL-LiteExpr}_0) \text{->isEmpty}(),$$

let p_{n+1} be an additional path item, and let

$$\text{OCL-LiteExpr}_{n+1} = p_{n+1} \cdot \text{OCL-LiteExpr}_n.$$

Then $\text{OCL-LiteExpr}_{n+1}^\dagger = \neg p_n^\dagger \cdot (\neg\text{OCL-LiteExpr}_n^\dagger)$,

Considering that $\neg C$ is equivalent to C , the \mathcal{ALCI} concept corresponding to OCL-LiteExpr has the form

$$\neg(p_n^\dagger \cdot (p_{n-1}^\dagger \cdot (\dots (p_1^\dagger \cdot (\text{OCL-LiteExpr}_0^\dagger)) \dots)))$$

Intuitively, this concept represents the fact that OCL-LiteExpr evaluates to true, for a given instance o , if o is not related through the path $p_n \dots p_1$ to any object that satisfies the condition specified by OCL-LiteExpr_0 . As in the previous case, if there is no select operation in the OCL expression, i.e., the OCL expression has the form $p_n \dots p_1 \text{->isEmpty}()$, then $\text{OCL-LiteExpr}_0^\dagger = \top$, and the constraint is encoded as $\neg(p_n^\dagger \cdot (p_{n-1}^\dagger \cdot (\dots (p_1^\dagger \cdot \top) \dots)))$

As an example, let us consider constraint 3 in its normal form. The overall OCL-Lite expression is encoded in \mathcal{ALCI} as $\neg(\exists\text{Audits} \cdot (\neg\exists\text{SponsoredBy} \cdot \top))$, and the \mathcal{ALCI} assertion corresponding to the constraint is:

$$\text{Person} \sqsubseteq \neg\exists\text{Audits} \cdot \neg\exists\text{SponsoredBy} \cdot \top$$

3. $\text{OCL-LiteExpr} = \text{oclIsTypeOf}(C)$, $\text{OCL-LiteExpr} = \text{not oclIsTypeOf}(C)$

The \mathcal{ALCI} concept $\text{OCL-LiteExpr}^\dagger$ corresponding to these OCL-Lite expressions is respectively

$$C, \quad \text{and} \quad \neg C,$$

4. $\text{OCL-LiteExpr} = \text{OCL-LiteExpr}_1$ and OCL-LiteExpr_2

The corresponding \mathcal{ALCI} concept $\text{OCL-LiteExpr}^\dagger$ is

$$\text{OCL-LiteExpr}_1^\dagger \sqcap \text{OCL-LiteExpr}_2^\dagger$$

5. $\text{OCL-LiteExpr} = \text{OCL-LiteExpr}_1$ or OCL-LiteExpr_2

The corresponding \mathcal{ALCI} concept $\text{OCL-LiteExpr}^\dagger$ is

$$\text{OCL-LiteExpr}_1^\dagger \sqcup \text{OCL-LiteExpr}_2^\dagger$$

6. $OCL-LiteExpr = OCL-LiteExpr_1$ implies $OCL-LiteExpr_2$
 The corresponding \mathcal{ALCI} concept $OCL-LiteExpr^\dagger$ is

$$\neg OCL-LiteExpr_1^\dagger \sqcup OCL-LiteExpr_2^\dagger$$

To further illustrate the mapping, we apply it to some other constraints of our running example. For instance, consider the normalized expression of constraint 1. This constraint is of kind 1, and its subexpressions are respectively of kinds 4, 3, and 2. Applying the corresponding mapping rules we obtain:

$$\text{Person} \sqsubseteq \exists \text{Organizes} . (\text{CriticalEvent} \sqcap \neg \exists \text{SponsoredBy} . \top)$$

As an example of an OCL-Lite expression of kind 6 we have constraint 2. According to the mapping rule, its corresponding \mathcal{ALCI} expression is:

$$\text{Event} \sqsubseteq \neg \exists \text{HeldWith} . \top \sqcup \neg \text{CriticalEvent}$$

Constraint 5 is of kind 5 once normalized. We recursively apply the mapping rules to each part of the disjunction: rules 1 and 3 to the first subexpression, and rules 5, 2, 1 to the second subexpression. The resulting \mathcal{ALCI} expression is:

$$\begin{aligned} \text{Company} \sqsubseteq \exists \text{SponsoredBy}^- . \text{CriticalEvent} \sqcup \\ \exists \text{SponsoredBy}^- . (\neg \exists \text{HeldWith} . \top \sqcup \exists \text{HeldWith} . \neg \exists \text{SponsoredBy} . \top) \end{aligned}$$

The following theorem states that the mapping from OCL-Lite to \mathcal{ALCI} is correct (we refer to [14] for the proof).

Theorem 1 (Correctness of the OCL-Lite encoding). *Let Γ be a set of OCL-Lite constraints and \mathcal{K}_Γ its \mathcal{ALCI} encoding. Then, Γ is satisfiable if and only if \mathcal{K}_Γ is satisfiable.*

Concerning the complexity of reasoning over UML/OCL-Lite schemas we first notice that reasoning just over the UML diagrams as proposed in this paper is an NP-complete problem. Indeed, the UML language we consider here is a sub-language of ER_{bool} which was proved to be NP-complete in [3], and the very same complexity proof applies to the UML language we use.

Theorem 2 (Complexity of UML/OCL-Lite). *Checking the satisfiability of UML/OCL-Lite conceptual schemas is an EXPTIME-complete problem.*

The upper bound follows from the fact that the \mathcal{ALCI} encoding is correct, and that reasoning in \mathcal{ALCI} is in EXPTIME. The lower bound is established by reducing satisfiability of \mathcal{ALC} TBoxes, which is known to be EXPTIME-complete, to satisfiability of OCL-Lite constraints (see [14] for the proof).

References

1. Anastasakis, K., Bordbar, B., Georg, G., Ray, I.: On challenges of model transformation from UML to Alloy. *Software and Systems Modeling* 9(1), 69–86 (2010)

2. Artale, A., Calvanese, D., Ibáñez-García, A.: Full satisfiability of UML class diagrams. In: Proc. of the 29th Int. Conf. on Conceptual Modeling (ER 2010). LNCS, vol. 6412, pp. 317–331. Springer (2010)
3. Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyashev, M.: Reasoning over extended ER models. In: Proc. of the 26th Int. Conf. on Conceptual Modeling (ER 2007). LNCS, vol. 4801, pp. 277–292. Springer (2007)
4. Beckert, B., Keller, U., Schmitt, P.H.: Translating the Object Constraint Language into first-order predicate logic. In: Proc. of the VERIFY Workshop at Federated Logic Conferences (FLoC). pp. 113–123 (2002)
5. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. *Artificial Intelligence* 168(1-2), 70–118 (2005)
6. Brucker, A.D., Wolff, B. (eds.): *The HOL-OCL Book*. Swiss Federal Institute of Technology (2006)
7. Calvanese, D., De Giacomo, G.: Expressive description logics. In: Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.) *The Description Logic Handbook: Theory, Implementation, and Applications*. pp. 178–218. Cambridge University Press (2003)
8. Calvanese, D., Lenzerini, M., Nardi, D.: Unifying class-based representation formalisms. *J. of Artificial Intelligence Research (JAIR)* 11, 199–240 (1999)
9. Dupuy, S., Ledru, Y., Chabre-Peccoud, M.: An overview of RoZ: A tool for integrating UML and Z specifications. In: Proc. of the 12th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2000). LNCS, vol. 1789, pp. 417–430. Springer (2000)
10. Fillottrani, P.R., Franconi, E., Tessaris, S.: The new ICOM ontology editor. In: Proc. of the 19th Int. Workshop on Description Logic (DL 2006). CEUR Electronic Workshop Proceedings, <http://ceur-ws.org/>, vol. 189 (2006)
11. Hartmann, S.: Coping with inconsistent constraint specifications. In: Proc. of the 20th Int. Conf. on Conceptual Modeling (ER 2001). LNCS, vol. 2224, pp. 241–255. Springer (2001)
12. Kuhlmann, M., Hamann, L., Gogolla, M.: Extensive validation of OCL models by integrating SAT solvers into USE. In: Proc. of the 49th Int. Conf. on Technology of Object-Oriented Languages and Systems (TOOLS 2011). LNCS, vol. 6705, pp. 290–306. Springer (2011)
13. Lenzerini, M., Nobili, P.: On the satisfiability of dependency constraints in entity-relationship schemata. *Information Systems* 15(4), 453–461 (1990)
14. Queralt, A., Artale, A., Calvanese, D., Teniente, E.: OCL-Lite: Finite reasoning on UML/OCL conceptual schemas. *Data and Knowledge Engineering (DKE)* 73, 1–22 (2012)
15. Queralt, A., Teniente, E.: Verification and validation of UML conceptual schemas with OCL constraints. *ACM Transactions on Software Engineering and Methodology* 21(2), 13:1–13:41 (2012)
16. Rumbaugh, J., Jacobson, I., Booch, G.: *The Unified Modeling Language Reference Manual*. Addison Wesley Publ. Co. (1998)
17. Snook, C.F., Butler, M.J.: UML-B: Formal modeling and design aided by UML. *ACM Transactions on Software Engineering and Methodology* 15(1), 92–122 (2006)
18. Wahler, M., Basin, D., Brucker, A.D., Koehler, J.: Efficient analysis of pattern-based constraint specifications. *Software and Systems Modeling* 9(2), 225–255 (2010)

Query Rewriting under Extensional Constraints in DL-Lite

Riccardo Rosati

DIAG, Sapienza Università di Roma, Italy
`lastname@dis.uniroma1.it`

1 Introduction

The *DL-Lite* family of description logics [4, 2] is currently one of the most studied ontology specification languages. *DL-Lite* constitutes the basis of the OWL2 QL language [1], which is part of the standard W3C OWL2 ontology specification language. The distinguishing feature of *DL-Lite* is to identify ontology languages in which expressive queries, in particular, unions of conjunctive queries (UCQs), over the ontology can be efficiently answered. Therefore, query answering is the most studied reasoning task in *DL-Lite* (see, e.g., [16, 11, 8, 18, 7, 6, 5]).

The most common approach to query answering in *DL-Lite* is through query rewriting. This approach consists of computing a so-called *perfect rewriting* of the query with respect to a TBox: the perfect rewriting of a query q for a TBox \mathcal{T} is a query q' that can be evaluated on the ABox only and produces the same results as if q were evaluated on both the TBox and the ABox. This approach is particularly interesting in *DL-Lite*, because, for every UCQ q , query q' can be expressed in first-order logic (i.e., SQL), therefore query answering can be delegated to a relational DBMS, since it can be reduced to the evaluation of an SQL query on the database storing the ABox.

The shortcoming of the query rewriting approach is that the size of the rewritten query may be exponential with respect to the size of the original query. In particular, this is true when the rewritten query is in disjunctive normal form, i.e., is an UCQ. On the other hand, [6] shows the existence of polynomial perfect rewritings of the query in nonrecursive datalog.

However, it turns out that the disjunctive normal form is necessary for practical applications of the query rewriting technique, since queries of more complex forms, once translated in SQL, produce queries with nested subexpressions that, in general, cannot be evaluated efficiently by current DBMSs. So, while in some cases resorting to more compact and structurally more complex perfect rewritings may be convenient, in general this strategy does not solve the problem of arriving at an SQL expression that can be effectively evaluated on the database.

In this scenario, a very interesting way to limit the size of the rewritten UCQ has been presented in [13]. This approach proposes the use of the so-called *ABox dependencies* to optimize query rewriting in *DL-Lite_A*. ABox dependencies are inclusions between concepts and roles which are interpreted as integrity constraints over the ABox: in other words, the ABox is guaranteed to satisfy such constraints. For this reason, in this paper we also call ABox dependencies *extensional constraints*. In the presence of such constraints, the query answering process can be optimized, since this additional

knowledge about the extensions of concepts and roles in the ABox can be exploited for optimizing query answering. Intuitively, the presence of ABox dependencies acts in a complementary way with respect to TBox assertions: while the latter complicate query rewriting, the former simplify it, since they state that some of the TBox assertions are already satisfied by the ABox.

As explained in [13], ABox dependencies have a real practical interest, since they naturally arise in many applications of ontologies, and in particular in ontology-based data access (OBDA) applications, in which a DL ontology acts as a virtual global schema for accessing data stored in external sources, and such sources are connected through declarative mappings to the global ontology. It turns out that, in practical cases, many ABox dependencies may be (automatically) derived from the mappings between the ontology and the data sources.

In this paper, we present an approach that follows the ideas of [13]. More specifically, we present **Prexto**, an algorithm for computing a perfect rewriting of a UCQ in the description logic $DL-Lite_{\mathcal{A}}$. **Prexto** is based on the query rewriting algorithm **Presto** [16]: with respect to the previous technique, **Prexto** has been designed to fully exploit the presence of extensional constraints to optimize the size of the rewriting; moreover, differently from **Presto**, it also uses concept and role disjointness assertions, as well as role functionality assertions, to reduce the size of the rewritten query. As already observed in [13], the way extensional constraints interact with reasoning, and in particular query answering, is not trivial at all: e.g., [13] defines a complex condition for the deletion of a concept (or role) inclusion from the TBox due to the presence of extensional constraints. In our approach, we use extensional constraints in a very different way from [13], which uses such constraints to “deactivate” corresponding TBox assertions in the TBox: conversely, we are able to define significant query minimizations even for extensional constraints for which there exists no corresponding TBox assertions. Based on these ideas, we define the **Prexto** algorithm: in particular, we restructure and extend the **Presto** query rewriting algorithm to fully exploit the presence of extensional constraints. Finally, we show that the above optimizations allow **Prexto** to outperform the existing query rewriting techniques for $DL-Lite$ in practical cases. In particular, we compare **Prexto** with **Presto** and with the optimization presented in [13].

This paper is an extended abstract of [15].

2 Preliminaries

We assume the reader is familiar with the basics of DLs as well as with $DL-Lite_{\mathcal{A}}$ [12].

Given an ABox \mathcal{A} , we denote by $\mathcal{I}_{\mathcal{A}}$ the $DL-Lite_{\mathcal{A}}$ interpretation such that, for every concept instance assertion $C(a)$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ iff $C(a) \in \mathcal{A}$, for every role instance assertion $R(a, b)$, $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle^{\mathcal{I}} \in R^{\mathcal{I}}$ iff $R(a, b) \in \mathcal{A}$, and for every attribute instance assertion $U(a, b)$, $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle^{\mathcal{I}} \in U^{\mathcal{I}}$ iff $U(a, b) \in \mathcal{A}$.

A conjunctive query (CQ) q is an expression of the form $q(\mathbf{x}) \leftarrow \alpha_1, \dots, \alpha_n$, where \mathbf{x} is a tuple of variables, and every α_i is an atom whose predicate is a concept name or a role name or an attribute name, and whose arguments are either variables or constants, such that every variable occurring in \mathbf{x} also occurs in at least one α_i . The variables occurring in \mathbf{x} are called the *distinguished variables* of q , while the variables occurring in some α_i but not in \mathbf{x} are called the *existential variables* of q . The predicate q is called

the *predicate* of the query, and the number of elements of x is called the *arity* of q . A CQ is a *Boolean* CQ if it has no distinguished variables.

A union of conjunctive queries (UCQ) Q is a set of conjunctive queries of the same arity and having the same query predicate. A UCQ Q is a Boolean UCQ if every CQ belonging to Q is Boolean.

Given a CQ q of arity n , we denote by $q(c)$ the Boolean CQ obtained from q by replacing the distinguished variables in x with the constants in the n -tuple of constants c . Given a CQ q of arity n , the evaluation of q in \mathcal{I} , denoted by $eval(q, \mathcal{I})$, is the set of n tuples of constants c such that \mathcal{I} satisfies the first-order sentence $q(c)$. The evaluation of a UCQ Q in \mathcal{I} , denoted by $eval(Q, \mathcal{I})$, is the set $\bigcup_{q \in Q} eval(q, \mathcal{I})$. The set of *certain answers* to a UCQ Q over a *DL-Lite_A* ontology $\langle \mathcal{T}, \mathcal{A} \rangle$, denoted by $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle)$, is the set of tuples $\bigcap_{\mathcal{I} \in Mod(\langle \mathcal{T}, \mathcal{A} \rangle)} eval(Q, \mathcal{I})$.

Given a UCQ Q and a TBox \mathcal{T} , a UCQ Q' is a *perfect rewriting* of Q with respect to \mathcal{T} if, for every ABox \mathcal{A} such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = eval(Q', \mathcal{I}_{\mathcal{A}})$. The above notion of perfect rewriting immediately extends to any query language for which the evaluation *eval* of queries on a first-order interpretation is defined. We remark that many algorithms are available to compute perfect rewritings in *DL-Lite* logics (e.g., [4, 12, 16, 11, 7, 6]).

In the following, for ease of exposition, we will not consider attributes in *DL-Lite_A* ontologies. However, all the algorithms and results that we present in this paper can be immediately extended to handle attributes (since attributes can essentially be treated in a way analogous to roles).

3 Extensional Constraints

We now define the notion of EBox, which constitutes a set of extensional constraints, i.e., constraints over the ABox. The idea of EBox has been originally introduced in [13], under the name of *ABox dependencies*.

The following definitions are valid for every DL, under the assumption that the assertions are divided into extensional assertions and intensional assertions, and extensional assertions correspond to atomic instance assertions.

Given a set of intensional assertions \mathcal{N} and an interpretation \mathcal{I} , we say that \mathcal{I} *satisfies* \mathcal{N} if \mathcal{I} satisfies every assertion in \mathcal{N} .

An *extensional constraint box*, or simply *EBox*, is a set of intensional assertions. Notice that, from the syntactic viewpoint, an EBox is identical to a TBox. Therefore, entailment of an assertion ϕ with respect to an EBox \mathcal{E} (denoted by $\mathcal{E} \models \phi$) is defined exactly in the same way as in the case of TBoxes.

Given an ABox \mathcal{A} and an EBox \mathcal{E} , we say that \mathcal{A} is *valid for* \mathcal{E} if $\mathcal{I}_{\mathcal{A}}$ satisfies \mathcal{E} .

Definition 1. (Admissible ABox) *Given a TBox \mathcal{T} and an EBox \mathcal{E} , an ABox \mathcal{A} is an admissible ABox for \mathcal{T} and \mathcal{E} if \mathcal{A} is consistent with \mathcal{T} and \mathcal{A} is valid for \mathcal{E} . We denote with $ADM(\mathcal{T}, \mathcal{E})$ the set of ABoxes \mathcal{A} that are admissible for \mathcal{T} and \mathcal{E} .*

Informally, an EBox acts as a set of *integrity constraints* over the ABox. Differently from other recent approaches that have proposed various forms of integrity constraints for DL ontologies (e.g., [9, 17]), an EBox constrains the ABox while totally discarding the TBox, since the notion of validity with respect to an EBox only considers the ABox.

We are now ready to define the notion of perfect rewriting in the presence of both a TBox and an EBox.

Definition 2. (Perfect rewriting in the presence of an EBox) Given a TBox \mathcal{T} , an EBox \mathcal{E} , and a UCQ Q , a FOL query ϕ is a perfect rewriting of Q with respect to $\langle \mathcal{T}, \mathcal{E} \rangle$ if, for every ABox $\mathcal{A} \in ADM(\mathcal{T}, \mathcal{E})$, $\langle \mathcal{T}, \mathcal{A} \rangle \models Q$ iff $\mathcal{I}_{\mathcal{A}} \models \phi$.

The above definition establishes a natural notion of perfect rewriting in the presence of an EBox \mathcal{E} . Since \mathcal{E} constrains the admissible ABoxes, the more selective is \mathcal{E} (for the same TBox \mathcal{T}), the more restricted the set $ADM(\mathcal{T}, \mathcal{E})$ is. If for instance, $\mathcal{E}, \mathcal{E}'$ are two EBoxes such that $\mathcal{E} \subset \mathcal{E}'$, we immediately get from the above definitions that $ADM(\mathcal{T}, \mathcal{E}) \supseteq ADM(\mathcal{T}, \mathcal{E}')$. Now, let Q be a UCQ, let ϕ be a perfect rewriting of Q with respect to $\langle \mathcal{T}, \mathcal{E} \rangle$ and let ϕ' be a perfect rewriting of Q with respect to $\langle \mathcal{T}, \mathcal{E}' \rangle$: ϕ will have to satisfy the condition $\langle \mathcal{T}, \mathcal{A} \rangle \models Q$ iff $\mathcal{I}_{\mathcal{A}} \models \phi$ for more ABoxes \mathcal{A} than query ϕ' . Consequently, ϕ will have to be a more complex query than ϕ' . Therefore, larger EBoxes in principle allow for obtaining simpler perfect rewritings.

As already explained, the goal of this paper is to use extensional constraints to optimize query rewriting in $DL\text{-}Lite_{\mathcal{A}}$. An intuitive explanation of how extensional constraints allow for simplifying query rewriting can be given by the following very simple example. Suppose we are given a TBox $\{Student \sqsubseteq Person\}$, an empty EBox \mathcal{E}_0 , and an EBox $\mathcal{E}_1 = \{Student \sqsubseteq Person\}$. Now, given a query $q(x) \leftarrow Person(x)$, a perfect rewriting of this query with respect to $\langle \mathcal{T}, \mathcal{E}_0 \rangle$ is the UCQ $\{q(x) \leftarrow Person(x) \quad q(x) \leftarrow Student(x)\}$, while a perfect rewriting of query q with respect to $\langle \mathcal{T}, \mathcal{E}_1 \rangle$ is the query q itself. Namely, under the EBox \mathcal{E}_1 we can ignore the TBox concept inclusion $Student \sqsubseteq Person$, since it is already satisfied by the ABox.

However, as already explained in [13], we can not always ignore TBox assertions that also appear in the EBox (and are thus already satisfied by the ABox). For instance, let q be the query $q \leftarrow C(x)$. If the TBox \mathcal{T} contains the assertions $\exists R \sqsubseteq C$ and $D \sqsubseteq \exists R^-$ and the EBox \mathcal{E} contains the assertion $\exists R \sqsubseteq C$, we cannot ignore this last inclusion when computing a perfect rewriting of q (or when answering query q). In fact, suppose the ABox is $\{D(a)\}$: then $\mathcal{A} \in ADM(\mathcal{T}, \mathcal{E})$ and query q is entailed by $\langle \mathcal{T}, \mathcal{A} \rangle$. But actually q is not entailed by $\langle \mathcal{T}', \mathcal{A} \rangle$ where $\mathcal{T}' = \mathcal{T} - \mathcal{E}$. From the query rewriting viewpoint, a perfect rewriting of q with respect to \mathcal{T} is the UCQ $\{q \leftarrow C(x) \quad q \leftarrow R(x, y) \quad q \leftarrow D(y)\}$, while a perfect rewriting of q with respect to \mathcal{T}' is the query q itself. And of course, the ABox \mathcal{A} shows that this last query is not a perfect rewriting of q with respect to $\langle \mathcal{T}, \mathcal{E} \rangle$. Therefore, also when computing a perfect rewriting, we cannot simply ignore the inclusions of the TBox that are already satisfied by the ABox (i.e., that belong to the EBox).

The example above shows that we need to understand under which conditions we are allowed to use extensional constraints to optimize query rewriting.

4 Prexto

In this section we present the algorithm Prexto (Perfect Rewriting under EXTensional cOnstraints). Prexto makes use of the algorithm Presto, originally defined in [16], which computes a nonrecursive datalog program constituting a perfect rewriting of a

```

Algorithm Presto( $Q, \mathcal{T}$ )
Input: UCQ  $Q$ ,  $DL-Lite_R$  TBox  $\mathcal{T}$ 
Output: nr-datalog query  $Q'$ 
begin
   $Q' = \text{Rename}(Q)$ ;
   $Q' = \text{DeleteUnboundVars}(Q')$ ;
   $Q' = \text{DeleteRedundantAtoms}(Q', \mathcal{T})$ ;
   $Q' = \text{Split}(Q')$ ;
  repeat
    if there exist  $r \in Q'$  and ej-var  $x$  in  $r$ 
      such that  $\text{Eliminable}(x, r, \mathcal{T}) = \text{true}$  and  $x$  has not already been eliminated from  $r$ 
    then begin
       $Q'' = \text{EliminateEJVar}(r, x, \mathcal{T})$ ;
       $Q'' = \text{DeleteUnboundVars}(Q'')$ ;
       $Q'' = \text{DeleteRedundantAtoms}(Q'', \mathcal{T})$ ;
       $Q' = Q' \cup \text{Split}(Q'')$ 
    end
  until  $Q'$  has reached a fixpoint;
  for each OA-predicate  $p_\alpha^n$  occurring in  $Q'$ 
  do  $Q' = Q' \cup \text{DefineAtomView}(p_\alpha^n, \mathcal{T})$ 
end

```

Fig. 1. The original Presto algorithm [16].

UCQ Q with respect to a $DL-Lite_A$ TBox \mathcal{T} . The algorithm **Presto** is reported in Figure 1. We refer the reader to [16] for a detailed explanation of the algorithm. For our purposes, it suffices to remind that the program returned by **Presto** uses auxiliary datalog predicates, called ontology-annotated (OA) predicates, to represent every basic concept and basic role that is involved in the query rewriting. E.g., the basic concept B is represented by the OA-predicate p_B^1 , while the basic role R is represented by the OA-predicate p_R^2 , where the superscript represents the arity of the predicate (actually, to handle Boolean subqueries, also 0-ary OA-predicates, i.e., predicates with no arguments, are defined: we refer the reader to [16] for more details).

In the following, we modify the algorithm **Presto**. In particular, we make the following changes. First, the final **for each** cycle of the algorithm (cf. Figure 1) is not executed: i.e., the rules defining the OA-predicates are not added to the returned program. Second, the algorithm **DeleteRedundantAtoms** is modified to take into account the presence of disjointness assertions and role functionality assertions in the TBox. More precisely, the following simplification rules are added to algorithm **DeleteRedundantAtoms**(Q', \mathcal{T}) (in which we denote basic concepts by B, C , role names by R, S , and datalog rules by the symbol r):

1. if $p_R^2(t_1, t_2)$ and $p_S^2(t_1, t_2)$ occur in r and $\mathcal{T} \models R \sqsubseteq \neg S$, then delete r from Q' ;
2. if $p_B^1(t)$ and $p_C^1(t)$ occur in r and $\mathcal{T} \models B \sqsubseteq \neg C$, then delete r from Q' ;
3. if $p_R^2(t_1, t_2)$ and $p_C^1(t_1)$ occur in r and $\mathcal{T} \models \exists R \sqsubseteq \neg C$, then delete r from Q' ;
4. if p_α^0 and p_β^0 occur in r and $\mathcal{T} \models \alpha^0 \sqsubseteq \neg \beta^0$, then delete r from Q' ;
5. if $p_B^1(t)$ and p_α^0 occur in r and $\mathcal{T} \models B^0 \sqsubseteq \neg \alpha^0$, then delete r from Q' ;
6. if $p_R^2(t_1, t_2)$ and p_α^0 occur in r and $\mathcal{T} \models R^0 \sqsubseteq \neg \alpha^0$, then delete r from Q' ;

Algorithm $\text{Prexto}(Q, \mathcal{T}, \mathcal{E})$
Input: UCQ Q , $DL\text{-Lite}_{\mathcal{A}}$ TBox \mathcal{T} , $DL\text{-Lite}_{\mathcal{A}}$ EBox \mathcal{E}
Output: UCQ Q'

```

begin
   $P = \text{Presto}(Q, \mathcal{T})$ ;
   $P' = \emptyset$ ;
  for each OA-predicate  $P_R^2$  occurring in  $P$  do
     $\Phi = \text{MinimizeViews}(R, \mathcal{E}, \mathcal{T})$ ;
     $P' = P' \cup \{p_B^2(x, y) \leftarrow S(x, y) \mid S \text{ is a role name and } S \in \Phi\}$ 
       $\cup \{p_B^2(x, y) \leftarrow S(y, x) \mid S \text{ is a role name and } S^- \in \Phi\}$ ;
  for each OA-predicate  $P_B^1$  occurring in  $P$  do
     $\Phi = \text{MinimizeViews}(B, \mathcal{E}, \mathcal{T})$ ;
     $P' = P' \cup \{p_B^1(x) \leftarrow C(x) \mid C \text{ is a concept name and } C \in \Phi\}$ 
       $\cup \{p_B^1(x) \leftarrow R(x, y) \mid \exists R \in \Phi\} \cup \{p_B^1(x) \leftarrow R(y, x) \mid \exists R^- \in \Phi\}$ ;
  for each OA-predicate  $P_N^0$  occurring in  $P$  do
     $\Phi = \text{MinimizeViews}(N^0, \mathcal{E}, \mathcal{T})$ ;
     $P' = P' \cup \{p_N^0 \leftarrow C(x) \mid C \text{ is a concept name and } C^0 \in \Phi\}$ 
       $\cup \{p_N^0 \leftarrow R(x, y) \mid R \text{ is a role name and } R^0 \in \Phi\}$ ;
   $P'' = P \cup P'$ ;
   $Q' = \text{Unfold}(P'')$ ;
   $Q' = \text{DeleteRedundantAtoms}(Q', \mathcal{E})$ ;
  return  $Q'$ 
end

```

Fig. 2. The Prexto algorithm.

7. if $p_R^2(t_1, t_2)$ and $p_R^2(t_1, t'_2)$ (with $t_2 \neq t'_2$) occur in r and $(\text{funct } R) \in \mathcal{T}$, then, if t_2 and t'_2 are two different constants, then delete r from Q' ; otherwise, replace r with the rule $\sigma(r)$, where σ is the substitution which poses t_2 equal to t'_2 .

Analogous simplification rules (which can be immediately derived) hold when R, S are inverse roles in cases 1, 3 and 7.

Example 1. Let us show the effect of the new transformations added to $\text{DeleteRedundantAtoms}$ through two examples. First, suppose $\mathcal{T} = \{B \sqsubseteq \neg B', (\text{funct } R)\}$ and suppose r is the rule $q(x) \leftarrow p_B^1(y), p_R^2(x, y), p_R^2(x, z), p_{B'}^1(z)$. First, the above rule 7 of algorithm $\text{DeleteRedundantAtoms}$ can be applied, which transforms r into the rule $q(x) \leftarrow p_B^1(y), p_R^2(x, y), p_{B'}^1(y)$. Then, the above rule 2 of algorithm $\text{DeleteRedundantAtoms}$ can be applied, hence this rule is deleted from the program. Intuitively, this is due to the fact that this rule looks for elements belonging both to concept B and to concept B' , which is impossible because the disjointness assertion $B \sqsubseteq \neg B'$ is entailed by the TBox \mathcal{T} . Therefore, it is correct to delete the rule from the program. \square

From now on, when we speak about Presto we refer to the above modified version of the algorithm, and when we speak about $\text{DeleteRedundantAtoms}$ we refer to the above modified version which takes into account disjointness and functionality assertions.

The Prexto algorithm is defined in Figure 2. The algorithm is constituted of the following four steps:

Algorithm MinimizeViews($B, \mathcal{E}, \mathcal{T}$)

Input: basic concept (or basic role, or 0-ary predicate) B ,
 $DL\text{-}Lite_{\mathcal{A}}$ EBox \mathcal{E} , $DL\text{-}Lite_{\mathcal{A}}$ TBox \mathcal{T}

Output: set of basic concepts (or basic roles, or 0-ary predicates) Φ''

begin

$\Phi = \{B' \mid \mathcal{T} \models B' \sqsubseteq B\};$

$\Phi' = \emptyset;$

for each $B' \in \Phi$ **do**

if there exists $B'' \in \Phi$ such that $\mathcal{E} \models B' \sqsubseteq B''$ and $\mathcal{E} \not\models B'' \sqsubseteq B'$

then $\Phi' = \Phi' \cup \{B'\};$

$\Phi'' = \Phi - \Phi';$

while there exist $B, B' \in \Phi'$ such that $B \neq B'$ and $\mathcal{E} \models B \sqsubseteq B'$ and $\mathcal{E} \models B' \sqsubseteq B$

do $\Phi'' = \Phi'' - \{B'\};$

return Φ''

end

Fig. 3. The MinimizeViews algorithm.

1. the nonrecursive datalog program P is computed by executing the Presto algorithm. This program P is not a perfect rewriting of Q yet, since the definition of the intermediate OA-predicates is missing;
2. the program P' is then constructed (by the three **for each** cycles of the program). This program contains rules defining the intermediate OA-predicates, i.e., the concept and role assertions used in the program P . To compute such rules, the algorithm makes use of the procedure MinimizeViews, reported in Figure 3. This procedure takes as input a basic concept (respectively, a basic role) B and computes a minimal subset Φ'' of the set Φ of the subsumed basic concepts (respectively, subsumed basic roles) of B which *extensionally cover* the set Φ , as explained below.
3. then, the overall nonrecursive datalog program $P \cup P'$ is unfolded, i.e., turned into a UCQ Q' . This is realized by the algorithm Unfold which corresponds to the usual unfolding of a nonrecursive program;
4. finally, the UCQ Q' is simplified by executing the algorithm DeleteRedundantAtoms which takes as input the UCQ Q' and the EBox \mathcal{E} (notice that, conversely, the first execution of DeleteRedundantAtoms within the Presto algorithm uses the TBox \mathcal{T} as input).

Notice that the bottleneck of the whole process is the above step 3, since the number of conjunctive queries generated by the unfolding may be exponential with respect to the length of the initial query Q (in particular, it may be exponential with respect to the maximum number of atoms in a conjunctive query of Q). As shown by the following example, the usage of extensional constraints done at step 2 through the MinimizeViews algorithm is crucial to handle the combinatorial explosion of the unfolding.

Example 2. Let \mathcal{T} be the following $DL\text{-}Lite_{\mathcal{A}}$ TBox:

$Company \sqsubseteq \exists \text{givesHighSalaryTo}^-$	$FulltimeStudent \sqsubseteq Unemployed$
$\exists \text{givesHighSalaryTo}^- \sqsubseteq Manager$	$FulltimeStudent \sqsubseteq Student$
$Manager \sqsubseteq Employee$	$isBestFriendOf \sqsubseteq knows$
$Employee \sqsubseteq HasJob$	(funct $isBestFriendOf$)
$\exists \text{receivesGrantFrom} \sqsubseteq StudentWithGrant$	(funct $isBestFriendOf^-$)
$StudentWithGrant \sqsubseteq FulltimeStudent$	$HasJob \sqsubseteq \neg Unemployed$

Moreover, let E_1, \dots, E_4 be the following concept inclusions:

$$\begin{aligned} E_1 &= FulltimeStudent \sqsubseteq StudentWithGrant & E_3 &= HasJob \sqsubseteq Employee \\ E_2 &= \exists receivesGrantFrom \sqsubseteq StudentWithGrant & E_4 &= Manager \sqsubseteq Employee \end{aligned}$$

and let $\mathcal{E}_1 = \{E_1\}$, $\mathcal{E}_2 = \{E_1, E_2\}$, $\mathcal{E}_3 = \{E_1, E_2, E_3\}$, $\mathcal{E}_4 = \{E_1, E_2, E_3, E_4\}$. Finally, let q_0, q_1, q_2, q_3 be the following simple queries:

$$\begin{aligned} q_0(x) &\leftarrow Student(x) \\ q_1(x) &\leftarrow Student(x), knows(x, y), HasJob(y) \\ q_2(x) &\leftarrow Student(x), knows(x, y), HasJob(y), knows(x, z), Unemployed(z) \\ q_3(x) &\leftarrow Student(x), knows(x, y), HasJob(y), knows(x, z), Unemployed(z), \\ &\quad knows(x, w), Student(w) \end{aligned}$$

Let us focus on query q_1 and let us consider the empty EBox. In this case, during the execution of $Prexto(q_1, \mathcal{T}, \emptyset)$ the algorithm `MinimizeViews` simply computes the subsumed sets of *Student*, *knows*, *HasJob*, which are, respectively:

$$\begin{aligned} \text{MinimizeViews}(Student, \emptyset, \mathcal{T}) &= \\ &\quad \{Student, FulltimeStudent, StudentWithGrant, \exists receivesGrantFrom\} \\ \text{MinimizeViews}(knows, \emptyset, \mathcal{T}) &= \{knows, isBestFriendOf\} \\ \text{MinimizeViews}(HasJob, \emptyset, \mathcal{T}) &= \{HasJob, Employee, Manager, \exists givesHighSalaryTo^-\} \end{aligned}$$

Since two sets are constituted of four predicates and one is constituted of two predicates, the UCQ returned by the unfolding step in $Prexto(q_1, \mathcal{T}, \mathcal{E})$ contains 32 CQs. This is also the size of the final UCQ, since in this case no optimizations are computed by the algorithm `DeleteRedundantAtoms`, because both the disjointness assertion and the role functionality assertions of \mathcal{T} have no impact on the rewriting of query q_1 .

Conversely, let us consider the EBox \mathcal{E}_4 : during the execution of $Prexto(q_1, \mathcal{T}, \mathcal{E})$, we obtain the following sets from the execution of the algorithm `MinimizeViews`:

$$\begin{aligned} \text{MinimizeViews}(Student, \mathcal{E}_4, \mathcal{T}) &= \{Student, StudentWithGrant\} \\ \text{MinimizeViews}(knows, \mathcal{E}_4, \mathcal{T}) &= \{knows, isBestFriendOf\} \\ \text{MinimizeViews}(HasJob, \mathcal{E}_4, \mathcal{T}) &= \{Employee, \exists givesHighSalaryTo^-\} \end{aligned}$$

Thus, the algorithm `MinimizeViews` returns only two predicates for *Student* and only two predicates for *HasJob*. Therefore, the final unfolded UCQ is constituted of 8 CQs (since, as above explained, the final call to `DeleteRedundantAtoms` does not produce any optimization). \square

It is possible to prove correctness of `Prexto` [15], which in turn implies the following property, which states that the computational cost of `Prexto` is no worse than all known query rewriting techniques for *DL-Lite_A* which compute UCQs.

Theorem 1. *Prexto(Q, T, E) runs in polynomial time with respect to the size of T ∪ E, and in exponential time with respect to the maximum number of atoms in a conjunctive query in the UCQ Q.*

query	algorithm	$\mathcal{E} = \emptyset$	$\mathcal{E} = \mathcal{E}_1$	$\mathcal{E} = \mathcal{E}_2$	$\mathcal{E} = \mathcal{E}_3$	$\mathcal{E} = \mathcal{E}_4$
q_0	Presto+unfolding	4	4	4	4	4
q_0	TBox-min	4	4	4	4	4
q_0	Prexto-noEBox	4	4	4	4	4
q_0	Prexto-noDisj	4	3	2	2	2
q_0	Prexto-full	4	3	2	2	2
q_1	Presto+unfolding	32	32	32	32	32
q_1	TBox-min	32	32	32	32	32
q_1	Prexto-noEBox	32	32	32	32	32
q_1	Prexto-noDisj	32	24	16	12	8
q_1	Prexto-full	32	24	16	12	8
q_2	Presto+unfolding	256	256	256	256	256
q_2	TBox-min	256	256	256	256	256
q_2	Prexto-noEBox	224	224	224	224	224
q_2	Prexto-noDisj	256	144	64	48	32
q_2	Prexto-full	224	126	106	42	28
q_3	Presto+unfolding	2048	2048	2048	2048	2048
q_3	TBox-min	2048	2048	2048	2048	2048
q_3	Prexto-noEBox	1584	1584	1584	1584	1584
q_3	Prexto-noDisj	2048	864	256	192	128
q_3	Prexto-full	1584	708	188	141	94

Fig. 4. Comparison of query rewriting techniques on \mathcal{T} , \mathcal{E} and queries q_0, q_1, q_2, q_3 .

5 Comparison

We now compare the optimizations introduced by Prexto with some of the current techniques for query rewriting in *DL-Lite*. In particular, we consider the simple *DL-Lite_A* ontology of Example 2 and compare the size of the UCQ rewritings generated by the original Presto algorithm, the rewriting based on the TBox minimization technique TBox-min shown in [13], and the Prexto algorithm. To single out the impact of the different optimizations introduced by Prexto, we present three different execution modalities for Prexto: without considering the EBox (we call this modality Prexto-noEBox); (ii) without considering disjointness axioms and role functionality axioms in the TBox (we call this modality Prexto-noDisj); (iii) and considering all axioms both in the TBox and in the EBox (we call this modality Prexto-full). Moreover, we will consider different EBoxes of increasing size, to better illustrate the impact of the EBox on the size of the rewriting.

The table reported in Figure 4 shows the impact on rewriting queries q_0, q_1, q_2 and q_3 of: (i) the disjointness axiom and the functional role axioms in \mathcal{T} ; (ii) the EBoxes $\mathcal{E}_1, \dots, \mathcal{E}_4$. In the table, we denote by Presto+unfolding the UCQ obtained by unfolding the nonrecursive datalog program returned by the Presto algorithm, and denote by TBox-min the execution of Presto+unfolding which takes as input the TBox minimized by the technique presented in [13] using the extensional inclusions in the EBox. These two rows can be considered as representative of the state of the art in query rewriting in *DL-Lite* (with and without extensional constraints): indeed, due to the simple structure of the TBox and the queries, every existing UCQ query rewriting technique for plain *DL-Lite* ontologies (i.e., ontologies without EBoxes) would generate UCQs of size analogous to Presto+unfolding (of course, we are not considering the approaches where the ABox is preprocessed, in which of course much more compact query rewritings can be defined [8, 13]). The third column of the table displays the results when the empty EBox was considered, while the fourth, fifth, sixth, and seventh column respec-

tively report the results when the EBox E_1, E_2, E_3, E_4 , was considered. The numbers in these columns represent the size of the UCQ generated when rewriting the query with respect to the TBox \mathcal{T} and the EBox \mathcal{E} : more precisely, this number is the number of CQs which constitute the generated UCQ. The results obtained in the case of query q_1 have been explained in Example 2.

The results of Figure 4 clearly show that even a very small number of EBox axioms may have a dramatic impact on the size of the rewritten UCQ, and that this is already the case for relatively short queries (like query q_2): this behavior is even more apparent for longer queries like q_3 . In particular, notice that, even when only two extensional inclusions are considered (case $\mathcal{E} = \mathcal{E}_2$), the minimization of the UCQ is already very significant. Moreover, for the queries under examination, extensional inclusions are more effective than disjointness axioms and role functionality axioms on the minimization of the rewriting size.

The results also show that the technique presented in [13] for exploiting extensional inclusions does not produce any effect in this case. This is due to the fact that the extensional inclusions considered in our experiment do not produce any minimization of the TBox according to the condition expressed in [13]. Conversely, the technique for exploiting extensional constraints of **Prexto** is indeed effective. For instance, notice that this technique is able to use extensional constraints (like E_2 and E_3) which have no counterpart in the TBox, in the sense that such concept inclusions are not entailed by the TBox \mathcal{T} .

Finally, we remark that the above simple example shows a situation which is actually not favourable for the algorithm, since there are very few extensional constraints and short (or even very short) queries: nevertheless, the experimental results show that, even in this setting, our algorithm is able to produce very significant optimizations. Indeed, the ideas which led to the **Prexto** algorithm came out of a large OBDA project that our research group is currently developing with an Italian Ministry. In this project, several relevant user queries could not be executed by our ontology reasoner (Quonto [3]) due to the very large size of the rewritings produced. For such queries, the minimization of the rewriting produced by the usage of the **Prexto** optimizations is even more dramatic than the examples reported in the paper, because the queries are more complex (at least ten atoms) and the number of extensional constraints is larger than in the example.

6 Conclusions

In this paper we have presented a query rewriting technique for fully exploiting the presence of extensional constraints in a *DL-Lite_A* ontology. Our technique clearly proves that extensional constraints may produce a dramatic improvement of query rewriting, and consequently of query answering over *DL-Lite_A* ontologies.

We believe that the present approach can be extended in several directions. First, it would be extremely interesting to generalize the **Prexto** technique to ontology-based data access (OBDA), where the ABox is only virtually specified through declarative mappings over external data sources: as already mentioned in the introduction, in this scenario extensional constraints would be a very natural notion, since they could be

automatically derived from the mapping specification. Then, it would be very interesting to extend the usage of extensional constraints beyond *DL-Lite_A* ontologies: in this respect, a central question is whether existing query rewriting techniques for other description logics (e.g., [11, 14]) can be extended with optimizations analogous to the ones of *Prexto*. Finally, we plan to fully implement our algorithm within the *Quonto/Mastro* system [3] for *DL-Lite_A* ontology management, and to further compare *Prexto* with other query rewriting techniques for *DL-Lite* (e.g., [11, 5, 10]).

Acknowledgments This research has been partially supported by the ICT Collaborative Project ACSI (Artifact-Centric Service Interoperation), funded by the EU under FP7 ICT Call 5, 2009.1.2, grant agreement n. FP7-257593.

References

1. OWL 2 web ontology language profiles (2009), <http://www.w3.org/TR/owl-profiles/>
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: The *DL-Lite* family and relations. *J. of Artificial Intelligence Research* 36, 1–69 (2009)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The *Mastro* system for ontology-based data access. *Semantic Web J.* 2(1), 43–53 (2011)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning* 39(3), 385–429 (2007)
5. Chortaras, A., Trivela, D., Stamou, G.B.: Optimized query rewriting for OWL 2 QL. In: *CADE*. pp. 192–206 (2011)
6. Gottlob, G., Schwenck, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: *Proc. of DL 2011* (2011)
7. Kikot, S., Kontchakov, R., Zakharyashev, M.: On (in)tractability of OBDA with OWL2QL. In: *Proc. of DL 2011* (2011)
8. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to query answering in *DL-Lite*. In: *Proc. of KR 2010*. pp. 247–257 (2010)
9. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. *J. of Web Semantics* 7(2), 74–89 (2009)
10. Orsi, G., Pieris, A.: Optimizing query answering under ontological constraints. *PVLDB* 4(11), 1004–1015 (2011)
11. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. *J. of Applied Logic* 8(2), 186–209 (2010)
12. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. *J. on Data Semantics X*, 133–173 (2008)
13. Rodriguez-Muro, M., Calvanese, D.: Dependencies: Making ontology based data access work in practice. In: *Proc. of AMW 2011* (2011)
14. Rosati, R.: On conjunctive query answering in \mathcal{EL} . In: *Proc. of DL 2007*. *CEUR*, ceur-ws.org, vol. 250, pp. 451–458 (2007)
15. Rosati, R.: *Prexto*: Query rewriting under extensional constraints in *DL-Lite*. In: *Proc. of ESWC 2012* (2012), to appear
16. Rosati, R., Almatelli, A.: Improving query answering over *DL-Lite* ontologies. In: *Proc. of KR 2010*. pp. 290–300 (2010)
17. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: *Proc. of AAAI 2010* (2010)
18. Thomas, E., Pan, J.Z., Ren, Y.: *TrOWL*: Tractable OWL 2 reasoning infrastructure. In: *Proc. of ESWC 2010*. pp. 431–435 (2010)

Elimination of Complex RIAs without Automata

František Simančík

Department of Computer Science, University of Oxford, UK

Abstract. We present an algorithm that eliminates complex role inclusion axioms (RIAs) from a *SR_{OIQ}* ontology preserving all logical consequences not involving non-simple roles. Unlike other existing methods, our algorithm does not explicitly construct finite automata recognizing the languages generated by the RIAs. Instead, it is formulated as a recursive expansion of universal restrictions, similar to well-known encodings of transitivity axioms.

1 Introduction

Complex role inclusion axioms (RIAs) $R_1 \circ \dots \circ R_n \sqsubseteq R$ are an important feature by which the web ontology language OWL 2 [2], based on the description logic (DL) *SR_{OIQ}* [5], extends the earlier standard OWL DL. Unrestricted use of complex RIAs causes undecidability of the basic reasoning tasks already in case of fairly inexpressive DLs and modal logics such as *ALC* [1]; decidability is recovered by requiring that RIAs, when viewed as context-free grammar rules $R \rightarrow R_1 \dots R_n$, generate regular languages [3, 4, 7]. However, checking if a given set of RIAs has this property is already difficult; this problem is related to checking regularity of *pure* context-free grammars [10] (which do not distinguish terminal and non-terminal symbols) whose decidability appears to have been long open [7]. To avoid this difficulty, a stronger syntactic *regularity* condition that requires RIAs to be acyclic (apart from a few selected cases such as transitivity axioms) is imposed in *SR_{OIQ}*. This condition is easy to check and allows for an effective construction of the underlying finite automata [5].

The standard automata construction for *SR_{OIQ}* [5] is an inductive procedure that performs non-trivial manipulations such as taking the mirrored copy and the disjoint union of previously constructed automata. To implement the procedure directly, developers of OWL 2 reasoners have to either look for suitable third-party libraries that support such operations, or resort to writing their own automata library. In this paper, we present an algorithm that eliminates complex RIAs from a *SR_{OIQ}* ontology without explicitly using finite automata. Instead, our algorithm is formulated as a simple recursive expansion of universal restrictions, similar to well-known encodings of transitivity axioms, using acyclicity of RIAs directly to ensure termination. For this reason, we believe that our method might be easier to implement in practice. Furthermore, we illustrate that, by introducing new rules for handling universal restrictions, the tableau algorithm for *SR_{OIQ}* [5] can be modified to apply our elimination on the fly. In that case, no pre-construction due to complex RIAs is needed at all.

Our recursive expansion of universal restrictions is inspired by and directly simulates the recursion in the standard automata construction. Therefore, for many purposes, the result of our elimination algorithm can be regarded as equivalent to the standard automata-based encoding [6].

2 Preliminaries

2.1 The DL \mathcal{SROIQ}

For a gentle introduction to DLs we refer the readers to the DL primer [8]. In this section we merely recall the definition (syntax only) of \mathcal{SROIQ} [5], together with the notions of a regular RBox and of polarity of concept occurrence. We follow the approach of Shearer [11] in assuming that the set of simple roles is given in the signature and calling a RIA $w \sqsubseteq R$ complex iff R is non-simple (even if w is of length 1).

A signature $\Sigma = \langle \Sigma_S, \Sigma_R, \Sigma_C, \Sigma_I \rangle$ consists of mutually disjoint sets of *atomic roles* Σ_R , *atomic concepts* Σ_C , and *individuals* Σ_I , together with a distinguished subset $\Sigma_S \subseteq \Sigma_R$ of *simple atomic roles*. The set of *roles* (over Σ) is $\mathbf{R} := \Sigma_R \cup \{R^- \mid R \in \Sigma_R\}$; the set of *simple roles* is $\mathbf{S} := \Sigma_S \cup \{S^- \mid S \in \Sigma_S\}$. A *role chain* is an expression of the form $R_1 \cdot \dots \cdot R_n$ with $n \geq 1$ and each $R_i \in \mathbf{R}$. The function $\text{inv}(\cdot)$ is defined on roles by $\text{inv}(R) := R^-$ and $\text{inv}(R^-) := R$ where $R \in \Sigma_R$, and extended to role chains by $\text{inv}(R_1 \cdot \dots \cdot R_n) := \text{inv}(R_n) \cdot \dots \cdot \text{inv}(R_1)$.

The set \mathbf{C} of \mathcal{SROIQ} *concepts* (over Σ) is defined recursively as follows:

$$\mathbf{C} := \Sigma_C \mid \{\Sigma_I\} \mid (\mathbf{C} \sqcap \mathbf{C}) \mid (\mathbf{C} \sqcup \mathbf{C}) \mid \neg \mathbf{C} \mid \exists \mathbf{R}. \mathbf{C} \mid \forall \mathbf{R}. \mathbf{C} \mid \geq n \mathbf{S}. \mathbf{C} \mid \leq n \mathbf{S}. \mathbf{C} \mid \exists \mathbf{S}. \text{Self}.$$

A *role inclusion axiom* (RIA) is either a *simple RIA* of the form $S_1 \sqsubseteq S_2$ where $S_1, S_2 \in \mathbf{S}$, or a *complex RIA* of the form $w \sqsubseteq R$ where w is a role chain and $R \in \mathbf{R} \setminus \mathbf{S}$. A *role assertion* is an axiom of the form $\text{Ref}(R)$ (reflexivity), $\text{lrr}(S)$ (irreflexivity), $\text{Uni}(R)$ (universality), or $\text{Dis}(S_1, S_2)$ (role disjointness), where $R \in \mathbf{R}$ and $S_{(i)} \in \mathbf{S}$. Transitivity and symmetry must be expressed as $R \cdot R \sqsubseteq R$ and $\text{inv}(R) \sqsubseteq R$ respectively. An *RBox* is a finite set of RIAs and role assertions.

A *regular order* \prec is an irreflexive transitive binary relation on the set of roles \mathbf{R} satisfying $R_1 \prec R_2$ iff $\text{inv}(R_1) \prec \text{inv}(R_2)$. An RBox \mathcal{R} is \prec -*regular* if each RIA in \mathcal{R} is of one of the following forms:

- (R1) $R_1 \cdot \dots \cdot R_n \sqsubseteq R$ with $R_i \prec R$ for all $1 \leq i \leq n$;
- (R2) $R \cdot R_1 \cdot \dots \cdot R_n \sqsubseteq R$ with $R_i \prec R$ for all $1 \leq i \leq n$;
- (R3) $R_1 \cdot \dots \cdot R_n \cdot R \sqsubseteq R$ with $R_i \prec R$ for all $1 \leq i \leq n$;
- (R4) $R \cdot R \sqsubseteq R$;
- (R5) $\text{inv}(R) \sqsubseteq R$.

An RBox \mathcal{R} is *regular* if it is \prec -regular for some regular order \prec . For a regular RBox \mathcal{R} , let $\prec_{\mathcal{R}}$ be the intersection of all regular orders \prec such that \mathcal{R} is \prec -regular; the *depth* of \mathcal{R} is the maximal n for which there exists a sequence $R_1 \prec_{\mathcal{R}} \dots \prec_{\mathcal{R}} R_n$. It is easy to show that if \mathcal{R} is regular, then it is $\prec_{\mathcal{R}}$ -regular.

A *TBox* is a finite set of *general concepts inclusions* (GCIs) of the form $C \sqsubseteq D$ where $C, D \in \mathbf{C}$. To keep the presentation simple, we do not allow ABox assertions; these can be expressed as GCIs using nominals. A \mathcal{SROIQ} ontology (over Σ) is a pair $\mathcal{O} = \langle \mathcal{R}, \mathcal{T} \rangle$ where \mathcal{R} is a regular RBox and \mathcal{T} a TBox.

Polarities of occurrences of \mathcal{SROIQ} concepts in concepts and GCIs are defined inductively as follows: C occurs positively in C . If C occurs positively (resp. negatively) in C' , then C occurs positively (resp. negatively) in $C' \sqcap D$, $D \sqcap C'$, $C' \sqcup D$, $D \sqcup C'$, $\exists \mathbf{R}. C'$, $\forall \mathbf{R}. C'$, $\geq n \mathbf{R}. C'$, and $D \sqsubseteq C'$, and C occurs negatively (resp. positively) in $\neg C'$, $\leq n \mathbf{R}. C'$, and $C' \sqsubseteq D$.

2.2 Conservative Encodings

We use the framework of conservative extensions [9] to prove correctness of our encoding of RIAs. Let \mathcal{O} be an ontology over a signature $\Sigma = \langle \Sigma_S, \Sigma_R, \Sigma_C, \Sigma_I \rangle$, and let \mathcal{Q} be an ontology over a (not necessarily strictly) larger signature. Then \mathcal{Q} is a *conservative encoding* of \mathcal{O} (also \mathcal{Q} is conservative over \mathcal{O}) if

- (i) for every model \mathcal{I} of \mathcal{O} there exists a model \mathcal{J} of \mathcal{Q} such that \mathcal{I} and \mathcal{J} have the same domain and coincide on the interpretation of Σ_R , Σ_C , and Σ_I , and
- (ii) for every model \mathcal{J} of \mathcal{Q} there exists a model \mathcal{I} of \mathcal{O} such that \mathcal{I} and \mathcal{J} have the same domain and coincide on the interpretation of Σ_R , Σ_C , and Σ_I .

Since this definition is sensitive to Σ , to avoid ambiguity, we will assume that each ontology carries its signature with it, so that each ontology is over only one signature. The signature may, however, contain symbols not occurring in the ontology. Note that, unlike the standard notion of a conservative extension, the above notion of a conservative encoding does not require that \mathcal{Q} contains all axioms from \mathcal{O} .

We define a *simple-conservative encoding* analogously except that the models \mathcal{I} and \mathcal{J} are only required to coincide on Σ_S , Σ_C , and Σ_I . As observed by Lutz et al. [9], this model-theoretic notion of conservativity implies that the two ontologies entail the same consequences over Σ_S , Σ_C , and Σ_I . We will prove that our encoding of complex RIAs produces a simple-conservative encoding of the input ontology. Thus, in particular, the two ontologies have the same classification (ignoring the extra atomic concepts introduced in the encoding). Furthermore, by introducing new concept definitions $A \equiv C$ and $B \equiv D$ in the original ontology, one can check subsumptions even between concepts C and D which contain non-simple roles.

2.3 Languages Generated by RIAs

Each RIA $w \sqsubseteq R$ can be expressed equivalently as $\text{inv}(w) \sqsubseteq \text{inv}(R)$; to avoid having to keep this in mind, let $\mathcal{R}^c := \mathcal{R} \cup \{\text{inv}(w) \sqsubseteq \text{inv}(R) \mid w \sqsubseteq R \in \mathcal{R}\}$ be the *completion* of the RBox \mathcal{R} . Note that \mathcal{R} and \mathcal{R}^c are equivalent and \mathcal{R} is \prec -regular iff \mathcal{R}^c is \prec -regular.

The languages $L_{\mathcal{R}}(R)$ are defined inductively by (i) $R \in L_{\mathcal{R}}(R)$ for each role R , and (ii) if $R_1 \cdot \dots \cdot R_n \sqsubseteq R \in \mathcal{R}^c$ and $w_i \in L_{\mathcal{R}}(R_i)$ for all $1 \leq i \leq n$, then $w_1 \cdot \dots \cdot w_n \in L_{\mathcal{R}}(R)$. Intuitively, $L_{\mathcal{R}}(R)$ is the language generated from the role R by the grammar rules $\{R \rightarrow w \mid w \sqsubseteq R \in \mathcal{R}^c\}$. Horrocks et al. [5] showed that if \mathcal{R} is regular, then each $L_{\mathcal{R}}(R)$ is a regular language, the finite automata recognizing $L_{\mathcal{R}}(R)$ can be effectively constructed by induction over $\prec_{\mathcal{R}}$, and the size of these automata is at most exponential in the depth of \mathcal{R} .

An interpretation function $\cdot^{\mathcal{I}}$ is extended to the languages $L_{\mathcal{R}}(R)$ as follows:

$$L_{\mathcal{R}}(R)^{\mathcal{I}} = \{\langle x, y \rangle \mid \text{there exists } w \in L_{\mathcal{R}}(R) \text{ such that } \langle x, y \rangle \in w^{\mathcal{I}}\}. \quad (1)$$

One can easily prove by induction on the definition of $L_{\mathcal{R}}(R)$ that $w \in L_{\mathcal{R}}(R)$ implies $\mathcal{R} \models w \sqsubseteq R$. The following proposition is a direct consequence of this fact.

Proposition 1. *If $\mathcal{I} \models \mathcal{R}$, then $L_{\mathcal{R}}(R)^{\mathcal{I}} = R^{\mathcal{I}}$.*

3 Motivation

In this section we motivate and present (first approximation of) our RIA-elimination algorithm. For simplicity, we do not yet consider symmetry axioms (R5) in this section.

Many *SRIOQ* constructors are restricted to simple roles and therefore do not interact with complex RIAs. The key step in eliminating complex RIAs is to capture the propagation of universal restrictions over non-simple roles. Consider the following property:

$$\text{if } x \in (\forall R.C)^{\mathcal{I}} \text{ and } \langle x, y \rangle \in L_{\mathcal{R}}(R)^{\mathcal{I}}, \text{ then } y \in C^{\mathcal{I}}. \quad (2)$$

Every model \mathcal{I} of the RBox \mathcal{R} satisfies (2) simply because $L_{\mathcal{R}}(R)^{\mathcal{I}} = R^{\mathcal{I}}$ by Proposition 1, in which case (2) coincides with semantics of universal restrictions. The main idea behind all methods for dealing with complex RIAs (e.g., [4–6]) is to axiomatise (using a finite number of GCIs) the property (2) for all $\forall R.C$ occurring in the ontology, and use this axiomatisation to simulate the presence of complex RIAs from \mathcal{R} .

In the simplest case, when all RIAs in \mathcal{R} are of the form (R1), this can be achieved by a simple recursive expansion of all universal restrictions occurring in the ontology. To expand $\forall R.C$, for each RIA $R_1 \cdot \dots \cdot R_n \sqsubseteq R \in \mathcal{R}^c$ introduce the axiom

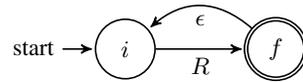
$$\forall R.C \sqsubseteq \forall R_1.\forall R_2 \dots \forall R_n.C, \quad (3)$$

and recursively expand all the nested universal restrictions on the right-hand side of (3). If all RIAs are of the form (R1), then $R_i \prec_{\mathcal{R}} R$ for each role R_i in (3), so the depth of the recursion is bounded by the depth of \mathcal{R} and the expansion terminates.

In case there are other forms of RIAs in \mathcal{R} , e.g., transitivity axioms, the recursion would never terminate. On the other hand, transitivity axioms can be eliminated using several well-known encodings. For example, to capture (2) for the concept $\forall R.C$ with respect to the transitivity axiom $R \cdot R \sqsubseteq R$, introduce two new atomic concepts I and F not in the signature of the the ontology and assert

$$\forall R.C \sqsubseteq I, \quad F \sqsubseteq C, \quad I \sqsubseteq \forall R.F, \quad F \sqsubseteq I. \quad (4)$$

This encoding is inspired by the fact that the RIA $R \cdot R \sqsubseteq R$ generates the regular language R^+ which is recognised by the following finite automaton:

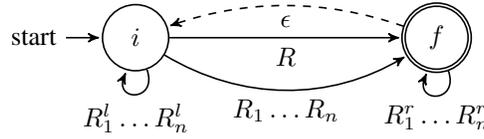


The encoding simulates the run of this automaton from all instances of $\forall R.C$ in a model of (4). Concepts I and F respectively correspond to the initial state i and the final state f , the first axiom initialises the automaton at all $x \in (\forall R.C)^{\mathcal{I}}$, the second axiom ensures that $y \in C^{\mathcal{I}}$ for all y in the final state, and the last two axioms encode the transitions of the automaton. This method generalises easily to all cases when the language $L_{\mathcal{R}}(R)$ is given by a finite automaton [6].

We propose a RIA-elimination algorithm that does not assume that the automaton is already fully constructed. Our method recursively expands all universal restrictions similarly to (3), but uses a two-state automaton at each step to handle the cyclic forms of RIAs similarly to (4). More specifically, to expand the universal restriction $\forall R.C$, introduce two new atomic concepts I and F , assert

- (E0) $\forall R.C \sqsubseteq I$, $F \sqsubseteq C$, and $I \sqsubseteq \forall R.F$,
- (E1) $I \sqsubseteq \forall R_1 \dots \forall R_n.F$ for each RIA $R_1 \dots R_n \sqsubseteq R \in \mathcal{R}^c$ of form (R1);
- (E2) $F \sqsubseteq \forall R_1^r \dots \forall R_n^r.F$ for each RIA $R \cdot R_1^r \dots R_n^r \sqsubseteq R \in \mathcal{R}^c$ of form (R2),
- (E3) $I \sqsubseteq \forall R_1^l \dots \forall R_n^l.I$ for each RIA $R_1^l \dots R_n^l \cdot R \sqsubseteq R \in \mathcal{R}^c$ of form (R3),
- (E4) $F \sqsubseteq I$ if $R \cdot R \sqsubseteq R \in \mathcal{R}^c$,

and recursively expand all the universal restrictions introduced in (E1)–(E3), but not the $\forall R.F$ introduced in (E0). Regularity of \mathcal{R} ensures that the depth of the recursion is bounded by the depth of \mathcal{R} . This encoding is inspired by the following automaton (the ϵ -edge is present iff $R \cdot R \sqsubseteq R \in \mathcal{R}^c$) used in the construction by Horrocks et al. [5] that recognises the language generated from R by the RIAs referred to in (E1)–(E4) :



Example 1. We will now demonstrate the RIA-elimination algorithm on an example. Let $\Sigma_S = \{P, R\}$, $\Sigma_R = \{P, R, S, T\}$, $\Sigma_C = \{A, B, C, D\}$, $\Sigma_I = \emptyset$, and let $\mathcal{O} = \langle \mathcal{R}, \mathcal{T} \rangle$ be the following ontology over the signature $\Sigma = \langle \Sigma_S, \Sigma_R, \Sigma_C, \Sigma_I \rangle$:

$$\mathcal{R} = \{T \cdot S \sqsubseteq T, \quad T \cdot T \sqsubseteq T, \quad R \cdot S \sqsubseteq S, \quad P \sqsubseteq R\}, \quad (5)$$

$$\mathcal{T} = \{A \sqsubseteq D \sqcup \forall T. \neg C, \quad B \sqsubseteq \exists T. \exists P. \exists S. C\}. \quad (6)$$

Note that \mathcal{R} is regular and $P \prec_{\mathcal{R}} R \prec_{\mathcal{R}} S \prec_{\mathcal{R}} T$. The RIA $P \sqsubseteq R$ is simple; the remaining RIAs in \mathcal{R} are complex. To expand the universal restriction $\forall T. \neg C$ occurring in \mathcal{T} , introduce new atomic concepts I_1 and F_1 , and assert the following axioms:

$$\forall T. \neg C \sqsubseteq I_1, \quad F_1 \sqsubseteq \neg C, \quad I_1 \sqsubseteq \forall T. F_1 \quad \text{by (E0)} \quad (7)$$

$$F_1 \sqsubseteq \forall S. F_1 \quad \text{by (E2) for } T \cdot S \sqsubseteq T \quad (8)$$

$$F_1 \sqsubseteq I_1 \quad \text{by (E4) for } T \cdot T \sqsubseteq T \quad (9)$$

Then, to recursively expand the new universal restriction $\forall S. F_1$ occurring in (8), introduce new atomic concepts I_2 and F_2 , and assert the following axioms:

$$\forall S. F_1 \sqsubseteq I_2, \quad F_2 \sqsubseteq F_1, \quad I_2 \sqsubseteq \forall S. F_2 \quad \text{by (E0)} \quad (10)$$

$$I_2 \sqsubseteq \forall R. I_2 \quad \text{by (E3) for } R \cdot S \sqsubseteq S \quad (11)$$

Finally, since we intend to keep all simple RIAs in the RBox and the role R is simple, the new universal restriction $\forall R. I_2$ introduced in (11) does not need to be further expanded. Let \mathcal{U} be the TBox consisting of the new axioms (7)–(11). The results of the next section will establish that the ontology $\mathcal{Q} = \langle \{P \sqsubseteq R\}, \mathcal{T} \cup \mathcal{U} \rangle$ is simple-conservative over \mathcal{O} , so, in particular, the two ontologies entail the same consequences over Σ_S , Σ_C , and Σ_I . For example, both \mathcal{O} and \mathcal{Q} entail $P \sqsubseteq R$ and $A \sqcap B \sqsubseteq D$. Note that this cannot be strengthened to all consequences over Σ since, for example, \mathcal{O} entails $T \cdot P \cdot S \sqsubseteq T$ and $B \sqsubseteq \exists T. C$, but \mathcal{Q} does not entail either of these axioms.

4 The RIA-Elimination Algorithm

In this section we formally present our RIA-elimination algorithm and prove that it produces a simple-conservative encoding of the input ontology. Some of the proofs are rather technical, in those cases we present only brief sketches here. More detailed proofs can be found in the appendix.

There are several important points about the algorithm that were omitted in the previous section in favour of simplicity; this is amended in this section. Firstly, if R is a symmetric role, i.e., $\text{inv}(R) \sqsubseteq R \in \mathcal{R}^c$, then the concept $\forall R.C$ is additionally expanded in the same way as $\forall \text{inv}(R).C$ would be. Secondly, since we do not assume that the ontology is in negation normal form, we have to treat negative occurrences of existential restrictions similarly to positive occurrences of universal restrictions. Finally, the expansion rule (E0) is inefficient because it introduces the axiom $\forall R.C \sqsubseteq I$ in which $\forall R.C$ occurs negatively. Negative occurrences of universal restrictions are not Horn, i.e., they lead to non-determinism in reasoning. To avoid this problem, instead of asserting $\forall R.C \sqsubseteq I$, the algorithm *replaces* all positive occurrences of $\forall R.C$ in the original ontology by I . This way we obtain a Horn-preserving encoding.

To keep track of the progress of the algorithm, we label those concepts $\forall R.C$ and $\exists R.C$ that still need to be expanded with \mathcal{R} as defined below.

Definition 1 (\mathcal{R} -labelled concepts). *Given an RBox \mathcal{R} , we introduce new concept constructors $\forall_{\mathcal{R}}R.C$ and $\exists_{\mathcal{R}}R.C$ called \mathcal{R} -labelled universals and \mathcal{R} -labelled existentials respectively. Their semantics is defined as follows:*

$$\begin{aligned} (\forall_{\mathcal{R}}R.C)^{\mathcal{I}} &= \{x \mid \forall y : \langle x, y \rangle \in L_{\mathcal{R}}(R)^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}, \\ (\exists_{\mathcal{R}}R.C)^{\mathcal{I}} &= \{x \mid \exists y : \langle x, y \rangle \in L_{\mathcal{R}}(R)^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}. \end{aligned}$$

\mathcal{R} -labelled concepts are *SRIOQ* concepts that may additionally contain \mathcal{R} -labelled universals and existentials. To distinguish them from normal *SRIOQ* concepts, we sometimes call the latter unlabelled. Similarly, we speak of \mathcal{R} -labelled (resp. unlabelled) ontologies.

Note that the semantics of \mathcal{R} -labelled concepts is irrelevant for the execution of the algorithm. It does, however, greatly simplify the proofs, since with this semantics we can prove that each intermediate expansion step of the algorithm already produces a simple-conservative encoding of the input ontology.

Given an input ontology $\mathcal{O} = \langle \mathcal{R}, \mathcal{T} \rangle$, the initial step of the algorithm is to remove all complex RIAs from \mathcal{O} (keeping all simple RIAs and all role assertions) and label all positive occurrences of universal restrictions and all negative occurrences of existential restrictions in \mathcal{O} with \mathcal{R} to indicate that they need to be expanded. This is defined more formally in the following two definitions.

Definition 2 (labelling). *Let \mathcal{R} be an RBox. For x an unlabelled concept or a TBox, let $\sigma_{\mathcal{R}}(x)$ be the result of labelling each positive occurrence of each universal restriction and each negative occurrence of each existential restriction in x with \mathcal{R} . Dually, let $\bar{\sigma}_{\mathcal{R}}(x)$ be the result of labelling each negative occurrence of each universal restriction and each positive occurrence of each existential restriction in x with \mathcal{R} .*

Definition 3 (initialisation). Let $\mathcal{O} = \langle \mathcal{R}, \mathcal{T} \rangle$ be an unlabelled *SRIOQ* ontology over a signature Σ . Let $\mathcal{R}^s := \mathcal{R} \setminus \{w \sqsubseteq R \in \mathcal{R} \mid w \sqsubseteq R \text{ is a complex RIA}\}$. The initialisation of \mathcal{O} is the \mathcal{R} -labelled ontology $\langle \mathcal{R}^s, \sigma_{\mathcal{R}}(\mathcal{T}) \rangle$ over the same signature Σ .

The next theorem proves that initialisation produces a simple-conservative encoding of \mathcal{O} . This captures the intuition that positive universal restrictions and negative existential restrictions are the only *SRIOQ* features that interact with complex RIAs.

Theorem 1. *The initialisation of \mathcal{O} is simple-conservative over \mathcal{O} .*

Proof (sketch). We must show that (i) for each model \mathcal{I} of \mathcal{O} there is a model \mathcal{J} of $\mathcal{Q} = \langle \mathcal{R}^s, \sigma_{\mathcal{R}}(\mathcal{T}) \rangle$ that agrees with \mathcal{I} on Σ_S , Σ_C , and Σ_I , and (ii) vice versa.

For (i), we show that each model \mathcal{I} of \mathcal{O} is already a model of \mathcal{Q} . Trivially, $\mathcal{I} \models \mathcal{R}$ implies $\mathcal{I} \models \mathcal{R}^s$ since $\mathcal{R}^s \subseteq \mathcal{R}$. By Proposition 1, we have $L_{\mathcal{R}}(R)^{\mathcal{I}} = R^{\mathcal{I}}$, so $(\forall R.C)^{\mathcal{I}} = (\forall_{\mathcal{R}} R.C)^{\mathcal{I}}$ and $(\exists R.C)^{\mathcal{I}} = (\exists_{\mathcal{R}} R.C)^{\mathcal{I}}$ for all concepts $\forall R.C$ and $\exists R.C$. This means that \mathcal{R} -labelling does not affect the interpretation of concepts in \mathcal{I} , so $\mathcal{I} \models \mathcal{T}$ implies $\mathcal{I} \models \sigma_{\mathcal{R}}(\mathcal{T})$. Therefore $\mathcal{I} \models \mathcal{Q}$.

For (ii), each model \mathcal{J} of \mathcal{Q} can be transformed to a model \mathcal{I} of \mathcal{O} by extending the interpretation of roles to $R^{\mathcal{I}} = L_{\mathcal{R}}(R)^{\mathcal{J}}$. From $\mathcal{J} \models \mathcal{R}^s$ one proves that \mathcal{I} and \mathcal{J} agree on simple roles. Checking that $\mathcal{I} \models \mathcal{R}$ is routine. The key step to prove $\mathcal{I} \models \mathcal{T}$ is to show by structural induction for each unlabelled concept D that $\sigma_{\mathcal{R}}(D)^{\mathcal{J}} \subseteq D^{\mathcal{I}} \subseteq \bar{\sigma}_{\mathcal{R}}(D)^{\mathcal{J}}$; since $\sigma_{\mathcal{R}}(\mathcal{T}) = \{C \sqsubseteq D \in \mathcal{T} \mid C \sqsubseteq D \in \mathcal{T}\}$, we can then infer $\mathcal{I} \models \mathcal{T}$ from $\mathcal{J} \models \sigma_{\mathcal{R}}(\mathcal{T})$. Hence $\mathcal{I} \models \mathcal{O}$. \square

After initialisation, the algorithm repeatedly expands all \mathcal{R} -labelled universals and existentials until it arrives at an unlabelled ontology. Before we define these expansions, in the next definition we first introduce an auxiliary function $\text{expand}(I \sqsubseteq \forall_{\mathcal{R}} R.F)$ that encodes the axiom $I \sqsubseteq \forall_{\mathcal{R}} R.C$ only using universals $\forall_{\mathcal{R}} S.D$ with $S \prec_{\mathcal{R}} R$. The function implements the transition function of the two-state automaton from the previous section, and additionally deals with symmetric roles R by expanding $I \sqsubseteq \forall_{\mathcal{R}} R.C$ in the same way as $I \sqsubseteq \forall_{\mathcal{R}} \text{inv}(R).C$. The correctness of this encoding is expressed in the following Proposition 2. Its proof can be found in the appendix.

Definition 4 (expand). Let \mathcal{R} be a regular *RBox*, R a role, and I and F atomic concepts. We define $\text{expand}'(I \sqsubseteq \forall_{\mathcal{R}} R.F)$ to be the set consisting of the following GCIs:

1. $I \sqsubseteq \forall R.F$,
2. $I \sqsubseteq \forall_{\mathcal{R}} R_1 \dots \forall_{\mathcal{R}} R_n.F$ for each $R_1 \dots R_n \sqsubseteq R \in \mathcal{R}^c$,
3. $F \sqsubseteq \forall_{\mathcal{R}} R_1 \dots \forall_{\mathcal{R}} R_n.F$ for each $R \cdot R_1 \dots R_n \sqsubseteq R \in \mathcal{R}^c$,
4. $I \sqsubseteq \forall_{\mathcal{R}} R_1 \dots \forall_{\mathcal{R}} R_n.I$ for each $R_1 \dots R_n \cdot R \sqsubseteq R \in \mathcal{R}^c$,
5. $F \sqsubseteq I$ if $R \cdot R \sqsubseteq R \in \mathcal{R}^c$,

where each R_i is distinct from R . Finally, we define $\text{expand}(I \sqsubseteq \forall_{\mathcal{R}} R.F)$ to be

$$\begin{cases} \text{expand}'(I \sqsubseteq \forall_{\mathcal{R}} R.F) \cup \text{expand}'(I \sqsubseteq \forall_{\mathcal{R}} \text{inv}(R).F) & \text{if } \text{inv}(R) \sqsubseteq R \in \mathcal{R}^c, \\ \text{expand}'(I \sqsubseteq \forall_{\mathcal{R}} R.F) & \text{otherwise.} \end{cases}$$

Proposition 2. *If $\mathcal{I} \models \text{expand}(I \sqsubseteq \forall_{\mathcal{R}} R.F)$, then $\mathcal{I} \models I \sqsubseteq \forall_{\mathcal{R}} R.F$.*

We are now ready to define the expansions of \mathcal{R} -labelled concepts. Similarly to structural transformation, $\forall_{\mathcal{R}}$ -expansion uses atomic concepts I and F as new names for the concepts $\forall_{\mathcal{R}}R.C$ and C respectively, replaces all positive occurrences of $\forall_{\mathcal{R}}R.C$ by I , adds $F \sqsubseteq C$, and, instead of asserting $I \sqsubseteq \forall_{\mathcal{R}}R.F$, it uses $\text{expand}(I \sqsubseteq \forall_{\mathcal{R}}R.F)$ to encode the same property. $\exists_{\mathcal{R}}$ -expansion works similarly but it additionally uses the equivalence of $\exists R.I \sqsubseteq F$ with $I \sqsubseteq \forall \text{inv}(R).F$; it uses atomic concepts I and F as new names for the concepts C and $\exists_{\mathcal{R}}R.C$ respectively, adds $C \sqsubseteq I$, replaces all negative occurrences of $\exists_{\mathcal{R}}R.C$ by F , and, instead of asserting $\exists_{\mathcal{R}}R.I \sqsubseteq F$, it uses $\text{expand}(I \sqsubseteq \forall_{\mathcal{R}}\text{inv}(R).F)$ to encode the same property. More formal definitions follow.

Definition 5 (substitution). For concepts C_{old} and C_{new} , and x a concept or a TBox, let $\rho^+[C_{new}/C_{old}](x)$ resp. $\rho^-[C_{new}/C_{old}](x)$ be the result of simultaneously replacing each positive resp. negative occurrence of C_{old} in x by C_{new} .

Definition 6 ($\forall_{\mathcal{R}}$ - and $\exists_{\mathcal{R}}$ -expansions). Let \mathcal{R} be a regular RBox and let $\mathcal{O} = \langle \mathcal{R}', \mathcal{T} \rangle$ be an \mathcal{R} -labelled ontology over a signature $\Sigma = \langle \Sigma_S, \Sigma_R, \Sigma_C, \Sigma_I \rangle$.

$\forall_{\mathcal{R}}$ -expansion:

Let $\forall_{\mathcal{R}}R.C$ be an \mathcal{R} -labelled universal occurring in \mathcal{O} . Let I and F be two different atomic concepts not in Σ_C . The $\forall_{\mathcal{R}}R.C$ -expansion of \mathcal{O} is the ontology

$$\langle \mathcal{R}', \rho^+[I/\forall_{\mathcal{R}}R.C](\mathcal{T}) \cup \{F \sqsubseteq C\} \cup \text{expand}(I \sqsubseteq \forall_{\mathcal{R}}R.F) \rangle.$$

$\exists_{\mathcal{R}}$ -expansion:

Let $\exists_{\mathcal{R}}R.C$ be an \mathcal{R} -labelled existential occurring in \mathcal{O} . Let I and F be two different atomic concepts not in Σ_C . The $\exists_{\mathcal{R}}R.C$ -expansion of \mathcal{O} is the ontology

$$\langle \mathcal{R}', \rho^-[F/\exists_{\mathcal{R}}R.C](\mathcal{T}) \cup \{C \sqsubseteq I\} \cup \text{expand}(I \sqsubseteq \forall_{\mathcal{R}}\text{inv}(R).F) \rangle.$$

In both cases, the resulting ontology is over the signature $\langle \Sigma_S, \Sigma_R, \Sigma_C \cup \{I, F\}, \Sigma_I \rangle$.

Note that the initialisation of an ontology contains only positive occurrences of \mathcal{R} -labelled universals and only negative occurrences of \mathcal{R} -labelled existentials. Furthermore, both expansions introduce only positive occurrences of \mathcal{R} -labelled universals. Therefore, when used in the context of the RIA-elimination algorithm, the above expansions will actually eliminate *all* occurrences of $\forall_{\mathcal{R}}R.C$ resp. $\exists_{\mathcal{R}}R.C$ from the ontology. The reason for making the substitutions polarity-sensitive was to make the following theorem hold for arbitrary ontologies.

Theorem 2. Let \mathcal{R} be a regular RBox, let \mathcal{O} be an \mathcal{R} -labelled ontology, and let \mathcal{Q} be a $\forall_{\mathcal{R}}R.C$ - or $\exists_{\mathcal{R}}R.C$ -expansion of \mathcal{O} . Then \mathcal{Q} is conservative over \mathcal{O} .

Proof (sketch). Let $\mathcal{O} = \langle \mathcal{R}', \mathcal{T}_1 \rangle$ be an \mathcal{R} -labelled ontology over Σ and let $\mathcal{Q} = \langle \mathcal{R}', \mathcal{T}_2 \rangle$ be a $\forall_{\mathcal{R}}R.C$ -expansion of \mathcal{O} . We need to show that (i) for each model \mathcal{I} of \mathcal{O} there exists a model \mathcal{J} of \mathcal{Q} that agrees with \mathcal{I} on Σ , and (ii) vice versa.

For (i), each model \mathcal{I} of \mathcal{O} can be extended to a model of \mathcal{Q} by interpreting the new concepts $I^{\mathcal{I}} := (\forall_{\mathcal{R}}R.C)^{\mathcal{I}}$ and $F^{\mathcal{I}} := (\exists_{\mathcal{R}}\text{inv}(R).\forall_{\mathcal{R}}R.C)^{\mathcal{I}}$. The substitution merely replaces some occurrences of $\forall_{\mathcal{R}}R.C$ by I , and $(\forall_{\mathcal{R}}R.C)^{\mathcal{I}} = I^{\mathcal{I}}$, so $\mathcal{I} \models \mathcal{T}_1$ implies $\mathcal{I} \models \rho^+[I/\forall_{\mathcal{R}}R.C](\mathcal{T}_1)$. To conclude that \mathcal{I} is a model of \mathcal{Q} , it remains to check that

\mathcal{I} satisfies each axiom in $\{F \sqsubseteq C\} \cup \text{expand}(I \sqsubseteq \forall_{\mathcal{R}} R.F)$. This is done using the definition of $I^{\mathcal{I}}$ and $F^{\mathcal{I}}$; for example, $\mathcal{I} \models F \sqsubseteq C$ holds because $\exists_{\mathcal{R}} \text{inv}(R). \forall_{\mathcal{R}} R.C \sqsubseteq C$ is a tautology. The remaining axioms can be checked similarly.

For (ii), we show that each model \mathcal{J} of \mathcal{Q} is already a model of \mathcal{O} . To prove $\mathcal{J} \models \mathcal{T}_1$, the key step is to prove by structural induction for all concepts D over Σ that

$$\rho^+[I / \forall_{\mathcal{R}} R.C](D)^{\mathcal{J}} \subseteq D^{\mathcal{J}} \subseteq \rho^-[I / \forall_{\mathcal{R}} R.C](D)^{\mathcal{J}}; \quad (12)$$

then $\mathcal{J} \models \rho^+[I / \forall_{\mathcal{R}} R.C](\mathcal{T}_1) \subseteq \mathcal{T}_2$ implies $\mathcal{J} \models \mathcal{T}_1$. The only non-trivial case in the induction is $D = \forall_{\mathcal{R}} R.C$, where (12) reduces to $I^{\mathcal{J}} \subseteq (\forall_{\mathcal{R}} R.C)^{\mathcal{J}}$; to show this, we apply Proposition 2 to $\mathcal{J} \models (\{F \sqsubseteq C\} \cup \text{expand}(I \sqsubseteq \forall_{\mathcal{R}} R.F)) \subseteq \mathcal{T}_2$ to infer $\mathcal{J} \models I \sqsubseteq \forall_{\mathcal{R}} R.C$, which is equivalent to the required $I^{\mathcal{J}} \subseteq (\forall_{\mathcal{R}} R.C)^{\mathcal{J}}$.

The proof of the case when \mathcal{Q} is an $\exists_{\mathcal{R}} R.C$ -expansion of \mathcal{O} , is similar: Each model \mathcal{I} of \mathcal{O} can be extended to a model of \mathcal{Q} by interpreting $I^{\mathcal{I}} := (\forall_{\mathcal{R}} \text{inv}(R). \exists_{\mathcal{R}} R.C)^{\mathcal{I}}$ and $F^{\mathcal{I}} := (\exists_{\mathcal{R}} R.C)^{\mathcal{I}}$. The substitution merely replaces some occurrences of $\exists_{\mathcal{R}} R.C$ by F , and $(\exists_{\mathcal{R}} R.C)^{\mathcal{I}} = F^{\mathcal{I}}$, so $\mathcal{I} \models \mathcal{T}_1$ implies $\mathcal{I} \models \rho^-[F / \exists_{\mathcal{R}} R.C](\mathcal{T}_1)$. To conclude that \mathcal{I} is a model of \mathcal{Q} , one can check that the definition of $I^{\mathcal{I}}$ and $F^{\mathcal{I}}$ satisfies each axiom in $\{C \sqsubseteq I\} \cup \text{expand}(I \sqsubseteq \forall_{\mathcal{R}} \text{inv}(R).F)$.

To show that each model \mathcal{J} of \mathcal{Q} is a model of \mathcal{O} , prove by structural induction for all concepts D over Σ that $\rho^-[F / \exists_{\mathcal{R}} R.C](D)^{\mathcal{J}} \subseteq D^{\mathcal{J}} \subseteq \rho^+[F / \exists_{\mathcal{R}} R.C](D)^{\mathcal{J}}$. This, in the only non-trivial case $D = \exists_{\mathcal{R}} R.C$, reduces to $(\exists_{\mathcal{R}} R.C)^{\mathcal{J}} \subseteq F^{\mathcal{J}}$; to show this, apply Proposition 2 to $\mathcal{J} \models (\{C \sqsubseteq I\} \cup \text{expand}(I \sqsubseteq \forall_{\mathcal{R}} \text{inv}(R).F)) \subseteq \mathcal{T}_2$ to infer $\mathcal{J} \models C \sqsubseteq \forall_{\mathcal{R}} \text{inv}(R).F$, which is equivalent to the required $(\exists_{\mathcal{R}} R.C)^{\mathcal{J}} \subseteq F^{\mathcal{J}}$. \square

The following theorem is our main result. It ensures that the RIA-elimination algorithm produces a simple-conservative encoding of the input ontology, and that the number of expansions is at most exponential in the depth of \mathcal{R} . Therefore, since each expansion is linear in the size of \mathcal{R} , the algorithm can be implemented to run in time exponential in the depth of \mathcal{R} , which is optimal since complex RIAs are known to incur an exponential increase in the complexity of reasoning [6].

Theorem 3. *Let $\mathcal{O} = \langle \mathcal{R}, \mathcal{T} \rangle$ be an unlabelled SROIQ ontology, let $\langle \mathcal{R}^s, \mathcal{T}_0 \rangle$ be the initialisation of \mathcal{O} , and let $(\mathcal{T}_i)_{i=1}^n$ be any sequence of TBoxes such that $\langle \mathcal{R}^s, \mathcal{T}_{i+1} \rangle$ is obtained from $\langle \mathcal{R}^s, \mathcal{T}_i \rangle$ by a $\forall_{\mathcal{R}}$ - or $\exists_{\mathcal{R}}$ -expansion. Then $\langle \mathcal{R}^s, \mathcal{T}_n \rangle$ is simple-conservative over \mathcal{O} . Moreover, n is bounded by $\|\mathcal{T}\| \cdot (2 \cdot \|\mathcal{R}\|)^d$ where d is the depth of \mathcal{R} .*

Proof. The proof uses the observation that if \mathcal{O}_1 is simple-conservative over \mathcal{O} , and \mathcal{O}_2 is conservative over \mathcal{O}_1 , then \mathcal{O}_2 is also simple-conservative over \mathcal{O} , which follows directly from the respective definitions. With this, it is easy to prove by induction on i that each $\langle \mathcal{R}^s, \mathcal{T}_i \rangle$ is simple-conservative over \mathcal{O} : the base case is established in Theorem 1, and the induction step follows from Theorem 2 and the above observation. Therefore, in particular, $\langle \mathcal{R}^s, \mathcal{T}_n \rangle$ is simple-conservative over \mathcal{O} .

To obtain the bound on n , let $r = 2 \cdot \|\mathcal{R}\|$. As remarked earlier, each $\forall_{\mathcal{R}} R.C$ - resp. $\exists_{\mathcal{R}} R.C$ -expansion eliminates all occurrences of $\forall_{\mathcal{R}} R.C$ resp. $\exists_{\mathcal{R}} R.C$ from the ontology, and introduces at most r new concepts $\forall_{\mathcal{R}} S.D$ (the factor 2 is due to symmetric roles) all satisfying $S \prec_{\mathcal{R}} R$. Therefore, each $\forall_{\mathcal{R}} R.C$ and $\exists_{\mathcal{R}} R.C$ occurring in \mathcal{O} (of which there are at most $\|\mathcal{T}\|$) can altogether generate at most $1 + r + r^2 + \dots + r^{d-1} < r^d$ \mathcal{R} -labelled universals and existentials, which yields the required bound of $\|\mathcal{T}\| \cdot r^d$. \square

I -intro: if $\forall R.C \in \mathcal{L}(x)$, then $\mathcal{L}(x) += I[\forall R.C]$, and $\mathcal{L}(x) += I[\forall \text{inv}(R).C]$ if $\text{inv}(R) \sqsubseteq R \in \mathcal{R}^c$.
F -intro: if $I[\forall R.C] \in \mathcal{L}(x)$ and $(R \in \mathcal{L}(x, y)$ or $\text{inv}(R) \in \mathcal{L}(y, x)$), then $\mathcal{L}(y) += F[\forall R.C]$.
F -elim: if $F[\forall R.C] \in \mathcal{L}(y)$, then $\mathcal{L}(y) += C$.
I -exp: if $I[\forall R.C] \in \mathcal{L}(x)$, then $\mathcal{L}(x) += \forall R_1 \dots \forall R_n.F[\forall R.C]$ for each $R_1 \dots R_n \sqsubseteq R \in \mathcal{R}^c$, and $\mathcal{L}(x) += \forall R_1 \dots \forall R_n.I[\forall R.C]$ for each $R \cdot R_1 \dots R_n \sqsubseteq R \in \mathcal{R}^c$.
F -exp: if $F[\forall R.C] \in \mathcal{L}(y)$, then $\mathcal{L}(y) += \forall R_1 \dots \forall R_n.F[\forall R.C]$ for each $R_1 \dots R_n \cdot R \sqsubseteq R \in \mathcal{R}^c$, and $\mathcal{L}(y) += I[\forall R.C]$ if $R \cdot R \sqsubseteq R \in \mathcal{R}^c$.

Fig. 1. Tableau rules for expansion of complex RIAs

Finally, as already observed in Example 1, the algorithm can be optimised by replacing all concepts $\forall_{\mathcal{R}}S.C$ and $\exists_{\mathcal{R}}S.C$ with a simple role S directly by $\forall S.C$ and $\exists S.C$ respectively, omitting their expansion. Interestingly, this makes the algorithm work without further modifications even in the presence of arbitrary cyclic simple RIAs; it is enough that complex RIAs are acyclic to ensure termination.

5 Elimination of Complex RIAs in the Tableau Algorithm

In this section we briefly sketch how the tableau algorithm for $SR\mathcal{OIQ}$ [5] can be modified to perform our encoding of complex RIAs on the fly. We assume that the readers are already familiar with the tableau algorithm. We use the standard notation $\mathcal{L}(x)$ and $\mathcal{L}(x, y)$ for labels of nodes and edges in the completion graph. We assume that with each concept $\forall R.C$ we can uniquely associate new concepts $I[\forall R.C]$ and $F[\forall R.C]$; these will be used in the expansion of $\forall R.C$. Since the tableau algorithm operates with concepts in negation normal form, it can never encounter a negative occurrence of an existential restriction, therefore expansion is only applicable to universal restrictions.

To obtain the modified tableau algorithm, replace all rules relating to universal restrictions (rules $\forall_1, \forall_2, \forall_3$ in [5]) by the rules in Fig. 1, where $\mathcal{L}(x) += C$ is a shorthand for $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C\}$ and each R_i is implicitly distinct from R . These rules can be readily compared to the expansion rules (E0)–(E4) from Section 3: rules I -intro, F -intro, and F -elim implement the axioms $\forall R.C \sqsubseteq I$, $I \sqsubseteq \forall R.F$, and $F \sqsubseteq C$ of (E0), rule I -exp implements the expansions (E1) and (E3), and rule F -exp implements the expansions (E2) and (E4). The rules can be extended with blocking conditions as usual.

Note that the first three rules together subsume the standard \forall -rule, but additionally introduce $I[\forall R.C]$ in $\mathcal{L}(x)$ and $F[\forall R.C]$ in $\mathcal{L}(y)$, even in case there are no complex RIAs in the ontology at all. To eliminate this overhead, similarly to the optimisation above, one can restrict rule I -intro to non-simple roles R , and apply the standard \forall -rule to universal restrictions with simple roles.

6 Conclusions

We presented an algorithm that encodes complex RIAs in *SR_QIQ* without constructing finite automata. The algorithm can also be applied in weaker DLs: apart from GCIs involving atomic concepts and concepts already occurring in the ontology, the algorithm introduces only GCIs of the form $I \sqsubseteq \forall R.F$ where I and F are atomic concepts, and R a possibly inverse role. Inverse roles are not strictly required either: if desired, each $I \sqsubseteq \forall R^-.F$ with an inverse role R^- can be replaced by the equivalent $\exists R.I \sqsubseteq F$.

Our algorithm shares many theoretical properties of the traditional approaches based on automata, e.g., it is Horn-preserving and runs in time exponential in the depth of the RBox. On the other hand, a notable difference between the two approaches is that, in the automata construction, one can apply standard techniques for minimising the number of automata states and thus potentially reduce the number of new concepts introduced in the encoding. While it might be difficult to provide similarly robust optimisation for our algorithm, several simple optimisations, such as the one presented here that restricts expansion of universal restrictions to non-simple roles, might already help in many realistic cases. Experimental evaluation of the algorithm is left for future work.

References

1. Baldoni, M., Giordano, L., Martelli, A.: A tableau calculus for multimodal logics and some (un)decidability results. In: Proc. of the Int. Conf. on Automatic Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 1998). pp. 44–59. Springer (1998)
2. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. *Journal of Web Semantics* 6(4), 309–322 (2008)
3. Demri, S., Nivelle, H.D.: Deciding regular grammar logics with converse through first-order logic. *Journal of Logic, Language and Information* 14(3), 289–329 (2005)
4. Goré, R., Nguyen, L.A.: A tableau calculus with automaton-labelled formulae for regular grammar logics. In: Proc. of the Int. Conf. on Automatic Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2005). pp. 138–152. Springer (2005)
5. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SR_QIQ*. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006). pp. 57–67. AAAI Press (2006)
6. Kazakov, Y.: *RI_Q* and *SR_QIQ* are harder than *SH_QIQ*. In: Proc. of the 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2008). pp. 274–284. AAAI Press (2008)
7. Kazakov, Y.: An extension of complex role inclusion axioms in the description logic *SROIQ*. In: Proc. of the 5th Int. Joint Conf. on Automated Reasoning (IJCAR 2010). pp. 472–486. Springer (2010)
8. Krötzsch, M., Simančík, F., Horrocks, I.: A description logic primer. CoRR abs/1201.4089 (2012), <http://arxiv.org/abs/1201.4089>
9. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007). pp. 453–458. AAAI Press (2007)
10. Maurer, H.A., Salomaa, A., Wood, D.: Pure grammars. *Information and Control* 44(1), 47–72 (1980)
11. Shearer, R.: Scalable Reasoning for Description Logics. Ph.D. thesis, University of Oxford (2010)

Improved Algorithms for Module Extraction and Atomic Decomposition

Dmitry Tsarkov

tsarkov@cs.man.ac.uk
School of Computer Science
The University of Manchester
Manchester, UK

Abstract. In recent years modules have frequently been used for ontology development and understanding. This happens because a module captures all the knowledge an ontology contains in a given area, and often is much smaller than the whole ontology. One useful modularisation technique for expressive ontology languages is locality-based modularisation, which allows for fast (polynomial) extraction of modules.

In order to better understand the modular structure of an ontology, a technique called Atomic Decomposition can be used. It efficiently builds a structure representing all possible modules for an ontology, regardless of the modularisation algorithm adopted and without the need to compute an exponential number of modules, as in a naive approach. This structure may be used e.g., for quick extraction of modules, or to investigate dependencies between modules, and so on.

However, existing algorithms for both locality-based module extraction and atomic decomposition do not scale well. This happens mainly because of their global nature: each iteration always explores the whole ontology, even when it is not necessary.

We propose algorithms for locality-based module extraction and atomic decomposition that work only on the relevant part of the ontology. This improves performance of algorithms by avoiding unnecessary checks. Empirical evaluation confirms a significant speed up on real-life ontologies.

1 Introduction

Following the great success of the OWL 2 family of ontology languages, they have become widespread as a knowledge representation formalism. This success, in particular, is based on reasoning facilities available for these languages, that are provided by a number of tools. However, the tools (usually) do not scale well. The worst-case computational complexity of the most expressive decidable fragment of OWL, OWL 2 DL, is N2EXPTIME. So there is a need to deal with large ontologies in an efficient manner.

One way of dealing with this issue during ontology development is to use *modules*. A module is a subset of an ontology that captures all the knowledge the ontology contains about a given set of terms. When reusing an existing ontology, instead of importing all of it to use a few terms and axioms, one could

extract a module based on a given set of terms, thus limiting the growth of the ontology that will be fed to the reasoner.

To do so, a module extraction algorithm is necessary. The notion of *conservative extensions* [3] allows one to define a module w.r.t. a signature Σ as a minimal set that preserves all entailments over Σ . However, deciding whether a subset of an ontology is a module in this sense is a non-trivial task. Even for simple DL languages, it is double exponential in time, whereas for expressive languages, like OWL 2 DL, this problem is undecidable.

Locality-based modules is an alternative solution for efficient module extraction in expressive logics. Intuitively, an axiom is *local* w.r.t. a signature if it does not affect any entailment that uses only terms from that signature. So the approach is to keep in the module only those axioms that are non-local to a given signature (while extending the signature as more axioms are added to the module). The traditional modularisation algorithm follows this idea by traversing all the axioms in the ontology, checking their locality and adding them to a module if non-local, updating the signature accordingly and then repeating the traversal until no new entities are added to a signature.

It is easy to see that this approach, while having polynomial (quadratic) run-time, has some room for improvement. The addition of a single term to a signature could lead to re-checking locality of every axiom in the ontology, including those that have nothing to do with the change in the signature, i.e., are not touched by either the old or the new signature. In the approach proposed in this paper only the axioms that might become non-local after a change of signature are re-checked.

There might be cases where one wants to extract more than just a single module from an ontology. In order to explore the modular structure of the ontology, the *atomic decomposition* approach has recently been investigated [1, 5]. Atomic decomposition can be viewed as a compact representation of *all* the modules of an ontology. In this approach the notion of *atom* is introduced as a subset of an ontology, whose axioms always co-occur in a module (i.e., for each module, either all of the axioms are included in the module or none of them occurs in it). A dependency between atoms, which mirrors the subset relation between corresponding modules, is also described in the Atomic Decomposition approach.

The algorithm for building the atomic decomposition of an ontology is also rather straightforward. First, the module for a signature of every axiom is built. Then axioms with equivalent modules are combined into a single atom. After the set of atoms is known, their modules are explored to derive dependencies between atoms.

As in the module extraction case, the atomic decomposition algorithm can be improved. Taking account of the notion of a module in the atomic decomposition structure, it is possible to significantly reduce the search space for modules, as well as for dependency relation. Empirical evaluation on large real-life ontologies shows an increase in performance of up to 50 times.

The rest of this paper is organised as follows. In Section 2 some preliminary notions are defined. Section 3 contains the definition of the original module extraction algorithm, the analysis of its inefficiencies and the improved algorithm, together with a proof of its correctness. Similarly, in Section 4 the original and improved algorithms for the atomic decomposition are discussed. Results of the evaluation of the algorithms involving several real-life ontologies are presented in Section 5. Finally some conclusions are drawn in Section 6.

2 Preliminaries

We assume that the reader is familiar with the notion of OWL 2 axiom, ontology and entailments. An *entity* is a named element of the signature of an ontology. For an axiom α , we denote by $\tilde{\alpha}$ the signature of that axiom, i.e. the set of all entities in α . The same notation is also used for a set of axioms.

Definition 1 (Module). *Let O be an ontology and Σ be a signature. A subset M of the ontology is called a module of O w.r.t. Σ if $M \models \alpha \iff O \models \alpha$, for every axiom α with $\tilde{\alpha} \subseteq \Sigma$.*

One of the ways to build modules is to use *locality* of axioms.

Definition 2 (Semantic Locality). *An axiom α is called $\top(\perp)$ -local w.r.t a signature Σ if replacing all named entities in $\tilde{\alpha} \setminus \Sigma$ with \top (resp. \perp) makes that axiom a tautology. An axiom α is called a tautology if it is local w.r.t. $\tilde{\alpha}$. An axiom α is called global if it is non-local w.r.t. \emptyset .*

Note that checking the tautology of an axiom α is done by checking the entailment of α by the empty ontology, i.e., $\emptyset \models \alpha$. In order to avoid this check (which involves reasoning and might be expensive) the notion of *syntactic locality* was introduced in [2].

We are not giving a formal definition of syntactic locality here. This would be an unnecessary complication, as the algorithms will use the locality checker as a black box. The intuition behind the syntactic locality is that it tries to simulate the entailment check by exploring the axiom structure and making decisions about locality by propagating constant values through expressions.

Syntactic locality is sound in the sense that every syntactically local axiom is also semantically local. The converse is not true, however: some syntactically non-local axiom are semantically local. We assume that the locality checker provides a method `ISNONLOCAL(α)` that returns **true** iff the axiom α is non-local.

Definition 3 (Atomic Decomposition). *A set of axioms A is an atom of an ontology O , if for every module M of O , either $A \subseteq M$ or $A \cap M = \emptyset$. An atom A is dependent on B (written $B \preceq A$) if $A \subseteq M$ implies $B \subseteq M$, for every module M . An Atomic Decomposition of an ontology O is a graph $G = \langle S, \preceq \rangle$, where S is the set of all atoms of O .*

Algorithm 1 Original Modularity Algorithm [2]

```
1: function GETMODULE( $\Sigma, O$ )
2:    $M \leftarrow \emptyset, \Sigma_0 \leftarrow \emptyset$ 
3:   repeat
4:      $\Sigma_0 \leftarrow \Sigma$ 
5:     for  $\alpha \in O$  do
6:       if  $\alpha \notin M$  and ISNONLOCAL( $\alpha, \Sigma$ ) then
7:          $M \leftarrow M \cup \alpha$ 
8:          $\Sigma \leftarrow \Sigma \cup \tilde{\alpha}$ 
9:       end if
10:    end for
11:  until  $\Sigma \neq \Sigma_0$ 
12:  return  $M$ 
13: end function
```

3 Module Extraction Algorithms

The locality-based module extraction is based on the following theorem.

Theorem 1 (Locality-based Module [2]). *Let $M \subseteq O$ be two ontologies such that all axioms in $O \setminus M$ are local w.r.t. $\Sigma \cup \tilde{M}$. Then M is a module of O w.r.t. Σ*

This claim holds for all types of modules, as well as the locality checking approach. The original algorithm, based on this theorem, is described here as an Algorithm 1, and is implemented in, e.g., OWL API.¹

3.1 Original Module Extraction Algorithm

The algorithm starts from the empty module and then goes through all the axioms to check their locality. If an axiom α is non-local w.r.t. the current signature, it is added to the module. In addition, the signature is extended with the signature of α . The whole process is repeated until the signature reaches a fixpoint (line 11).

While having a simple structure and being easily understandable, the traditional algorithm has some inefficiencies. The most obvious one comes from the fact that it is necessary to check all the remaining axioms in the ontology if a single entity is added to the signature. This leads to the worst-case complexity of $O(n^2)$, where n is the number of axioms in the ontology. Indeed, if every run of the loop in lines 3–11 adds a single axiom to a module, increasing the signature on each step, the loop will be run n times and about $n^2/2$ locality checks will be made.

¹ <http://owlapi.sourceforge.org>

Algorithm 2 Improved Modularity Algorithm

```
1: function GETMODULE( $\Sigma, O$ )
2:    $SA \leftarrow \emptyset, Globals \leftarrow \emptyset, M \leftarrow \emptyset$ 
3:   for all  $\alpha \in O$  do                                      $\triangleright$  Initialize  $SA$  and  $Globals$ 
4:     if ISNONLOCAL( $\alpha, \emptyset$ ) then                        $\triangleright$  global axiom
5:        $Globals \leftarrow Globals \cup \{\alpha\}$ 
6:     else
7:       for all  $\sigma \in \tilde{\alpha}$  do
8:          $SA(\sigma) \leftarrow SA(\sigma) \cup \{\alpha\}$ 
9:       end for
10:    end if
11:  end for
12:   $S \leftarrow \Sigma$                                         $\triangleright$  Initialise working set
13:  for all  $\gamma \in Globals$  do                                $\triangleright$  Global axioms are always in the module
14:    ADDNONLOCAL( $\gamma, \Sigma, M, S$ )
15:  end for
16:  for all  $\sigma \in S$  do
17:     $S = S \setminus \{\sigma\}$ 
18:    for all  $\alpha \in SA(\sigma)$  do
19:      ADDNONLOCAL( $\alpha, \Sigma, M, S$ )
20:    end for
21:  end for
22:  return  $M$ 
23: end function

24: procedure ADDNONLOCAL( $\alpha, \Sigma, M, S$ )
25:  if  $\alpha \notin M$  and ISNONLOCAL( $\alpha, \Sigma$ ) then
26:     $M \leftarrow M \cup \alpha$ 
27:     $S \leftarrow S \cup (\tilde{\alpha} \setminus \Sigma)$ 
28:     $\Sigma \leftarrow \Sigma \cup \tilde{\alpha}$ 
29:  end if
30: end procedure
```

3.2 Improved Modularity Algorithm

The approach we propose in this paper replaces the global search over all axioms in the ontology with a search over a reduced set of possibly affected axioms. When a new entity is added to the signature, the algorithm checks locality only of the axioms that contain this entity in their signature. This is correct due to the following fact:

Proposition 1. *Let Σ be a signature, and α an axiom such that α is local w.r.t. Σ . Then α is also local w.r.t. any signature $\Sigma \cup \Sigma'$ such that $\Sigma' \cap \tilde{\alpha} = \emptyset$.*

Proof. Let $\alpha|_{\Sigma}^c$ denote the axiom α in which all entities not in Σ are replaced with c , where c is either \top or \perp depending on the locality type. Then the claim of the proposition follows from two simple observations:

1. $\alpha|_{\Sigma_1 \cup \Sigma_2}^c = (\alpha|_{\Sigma_1}^c)|_{\Sigma_2}^c$

$$2. \alpha|_{\Sigma \setminus \tilde{\alpha}}^c = \alpha$$

Since α is local w.r.t. Σ , $\alpha|_{\Sigma}^c$ is a tautology. The, by the first item above, $\alpha|_{\Sigma \cup \Sigma'}^c = (\alpha|_{\Sigma}^c)|_{\Sigma'} = (\alpha|_{\Sigma}^c)|_{\Sigma' \setminus \tilde{\alpha}}^c$, which, by the second item, equals to $\alpha|_{\Sigma'}^c$. So, $\alpha|_{\Sigma \cup \Sigma'}^c$ coincides with $\alpha|_{\Sigma'}^c$, i.e., is a tautology. Thus, α is local w.r.t. $\Sigma \cup \Sigma'$. \square

Algorithm 2 implements the proposed approach. In lines 3–11, the auxiliary structures for the algorithms are initialised. One of these structures is a map SA that associates each entity with the set of axioms containing it in their signature. Another is a set of global axioms $Globals$. Global axioms should be treated in a special way as they are part of every module independently of their signature.

After these structures are created, the algorithm initialises the working set S with the initial signature Σ . Then, all the global axioms from the set $Globals$ are added to the module using the `ADDNONLOCAL` procedure.

In the main cycle (lines 16–21) an entity σ is taken from the set S , then the set of affected axioms is retrieved using the map SA . Each of these axioms is checked for locality and, if non-local, is added to the module by `ADDNONLOCAL`.

The `ADDNONLOCAL` procedure is defined in the lines 24–30 of the Algorithm 2. If an axiom is found non-local, it is added to the module M , and its signature is added to Σ . Moreover, every new entity is added to the working set S (line 27) to allow further search for the axioms that are non-local w.r.t. the extended signature.

The correctness of the algorithm can be proved by induction on the size of the signature Σ . The basis of induction, for the empty signature: all that goes to the module is the set of global axioms, which is done in the lines 13–15 of the algorithm. Assume that for a given Σ all the necessary locality checks have been performed for axioms in the ontology O . Let us now check the case when a new entity σ is added to Σ . In this case all the axioms from O that contain Σ in their signature, are re-checked for locality w.r.t. new signature. All other axioms, according to Proposition 1, will keep their locality status, so there is no need to re-check them.

Note that the computational complexity of the improved algorithm is different from the one of the original one. Now every axiom α is checked at most $|\tilde{\alpha}|$ times, so the overall complexity is $O(N \times s)$, where N is the size of the ontology O and $s = \max_{\alpha \in O} (|\tilde{\alpha}|)$.

It is also worth noting that the initialisation of auxiliary structures (lines 3–11) can be done only once for every ontology, and then reused for consequent module extraction queries.

4 Atomic Decomposition Algorithms

Now let us introduce the algorithms for the atomic decomposition of an ontology. For the ease of explanation we assume that the ontology for the atomic decomposition does not contain tautologies (i.e. axioms that are local w.r.t. their own signature). They have no sense in the atomic decomposition, as they does not

Algorithm 3 Original Atomic Decomposition [1]

```
1: procedure ATOMICDECOMP( $O$ )
2:    $Gen \leftarrow \emptyset, Module \leftarrow \emptyset, Atom \leftarrow \emptyset, \preceq \leftarrow \emptyset$ 
3:   for all  $\alpha \in O$  do                                      $\triangleright$  build all atoms and modules
4:      $Module(\alpha) \leftarrow \text{GETMODULE}(\tilde{\alpha}, O)$ 
5:     if ISNEWMODULE( $\alpha, Gen$ ) then
6:        $Atom(\alpha) \leftarrow \{\alpha\}$ 
7:        $Gen \leftarrow Gen \cup \alpha$ 
8:     end if
9:   end for

10:  for all  $\alpha \in Gen$  do                                      $\triangleright$  build all dependencies
11:    for all  $\beta \in Gen$  do
12:      if  $\alpha \in Module(\beta)$  then
13:         $\preceq \leftarrow \preceq \cup \{Atom(\alpha), Atom(\beta)\}$ 
14:      end if
15:    end for
16:  end for
17: end procedure

18: function ISNEWMODULE( $\alpha, Gen$ )
19:  for all  $\beta \in Gen$  do
20:    if  $Module(\alpha) = Module(\beta)$  then
21:       $Atom(\beta) \leftarrow Atom(\beta) \cup \{\alpha\}$ 
22:    return false
23:    end if
24:  return true
25: end for
26: end function
```

belong to any (locality-based) module of the ontology. In order to achieve this one have to check all the axioms and remove the tautologies from the ontology.

4.1 The Original Atomic Decomposition Algorithm

The original atomic decomposition algorithm presented here was described by Del Vescovo et al [1]. It contains two independent parts. The first part (lines 3–9) builds all the atoms of the ontology. It is done by creating a module for a signature of every axiom α (line 4), and by comparing this module to already created modules. If such a module already exists in the ontology (which is checked in the auxiliary procedure ISNEWMODULE, line 20), then the axiom is added to the atom, represented by the already checked axiom β (line 21). If no module is equivalent to the module for α , then α goes to a new atom (line 6).

The second part of the algorithm, lines 10–16, builds the dependency relation \preceq . It goes through all atoms and sets the dependency between atoms A and B if an axiom from A is contained in the module built for axioms from the atom B .

4.2 Improved Atomic Decomposition Algorithm

As in the case of the module extraction algorithm, the traditional atomic decomposition algorithm suffers from some inefficiencies. The first is independence of module creation: all the modules are created, using the whole ontology as a starting point. However, in many cases a module is included into another module with a bigger signature. This is a consequence of the following observation.

Algorithm 4 Improved Atomic Decomposition

Require: An ontology O

Ensure: Set to atoms $Atom$, set of modules $Module$, dependency function \preceq

```

1: procedure BUILDAD( $O$ )
2:   for all  $\alpha \in O$  do
3:     if  $Atom(\alpha) = \emptyset$  then
4:       BUILDATOMSINMODULE( $\alpha, null$ )           ▷ Set  $Module(null)$  to be  $O$ 
5:     end if
6:   end for
7:   TRANSITIVECLOSE( $\preceq$ )
8: end procedure

9: function BUILDATOMSINMODULE( $\alpha, \beta$ )
10:  if  $Atom(\alpha) \neq \emptyset$  then                ▷ The atom for  $\alpha$  is already known
11:    return  $\alpha$ 
12:  end if
13:   $\delta \leftarrow$  GETATOMSEED( $\alpha, \beta$ )
14:   $Atom(\delta) \leftarrow Atom(\delta) \cup \{\alpha\}$ 
15:  if  $\delta = \beta$  then
16:    return  $\beta$ 
17:  end if
18:  for all  $\gamma \in Module(\alpha) \setminus \{\alpha\}$  do
19:     $\delta \leftarrow$  BUILDATOMSINMODULE( $\gamma, \alpha$ )
20:     $\preceq \leftarrow \preceq \cup \langle Atom(\delta), Atom(\alpha) \rangle$ 
21:  end for
22:  return  $\alpha$ 
23: end function

24: function GETATOMSEED( $\alpha, \beta$ )
25:   $Module(\alpha) \leftarrow$  GETMODULE( $\tilde{\alpha}, Module(\beta)$ )
26:  if  $Module(\alpha) = Module(\beta)$  then
27:    return  $\beta$ 
28:  else
29:    return  $\alpha$ 
30:  end if
31: end function

```

Proposition 2. *Let α, β be axioms in an ontology O . Let $Module(\gamma, O)$ denote the module of O w.r.t. $\tilde{\gamma}$. Then $Module(\beta, O) \subseteq Module(\alpha, O)$ whenever $\beta \in Module(\alpha, O)$.*

In fact, this proposition follows from *depleteness* of the locality-based modules [1, Proposition 2.2, claim iii]. So in order to build a module for β it is enough to explore only axioms in the module for α .

Another observation stems from the analysis of an dependency relation structure. Lines 12–13 of the Algorithm 3 imply that all atoms on which $Atom(\beta)$ depends are contained in $Module(\beta)$. But in this case there is no need to look outside that module for the dependencies. At the same time, the dependency relation could be build during the atom creation process.

These two ideas lie at the foundation of the improved atomic decomposition algorithm, presented as Algorithm 4. The main cycle (lines 2–6) ensures that an atom is built for every axiom, using the whole ontology as a starting point. After all atoms are created, the dependency relation is completed by using the standard transitive closure algorithm (line 7).

The main ingredient of the algorithm is implemented as a recursive function `BUILDATOMSINMODULE`. It takes two parameters: an axiom α , for which the atom (and module) are going to be built; and an axiom β , which is a “parent” of an α in the sense that $Module(\alpha) \subseteq Module(\beta)$. For special case $\beta = null$, as in line 4 of the code, we assume that $Module(\beta)$ is the whole ontology O . The function returns a representative of the $Atom(\alpha)$.

First, it checks whether an atom for α has been already created (lines 10–12). In this case there is nothing to do and α is returned as a representative.

Then, using the module of a parent axiom as a starting point, the module for α is created (line 25). Then, like in the function `ISNEWMODULE` from Algorithm 3, the algorithm checks whether such module already exists. However, unlike in function `ISNEWMODULE`, only one check is required here (line 26), namely, to compare it with the parent module. Then axiom α is added to an atom, obtained by function `GETATOMSEED` (line 14) and, if the atom already exists (i.e., is represented by the parent axiom β) then the parent is returned.

If the atom is new, i.e. $Module(\alpha) \neq Module(\beta)$, the algorithm recursively builds all atoms inside $Module(\alpha)$ (lines 18–21): `BUILDATOMSINMODULE` is called for all axioms in $Module(\alpha)$ with α as a parent. The dependency relation is updated accordingly (line 20), as $Atom(\alpha)$ depends on every atom in the $Module(\alpha)$. In the end, α as a representative of a new module, is returned.

5 Empirical Evaluation

The improved algorithms described in Sections 3.2 and 4.2 were implemented in the FaCT++ Description Logic reasoner [4]. We have done some experiments, which show considerable performance improvement over the original versions of the algorithms.

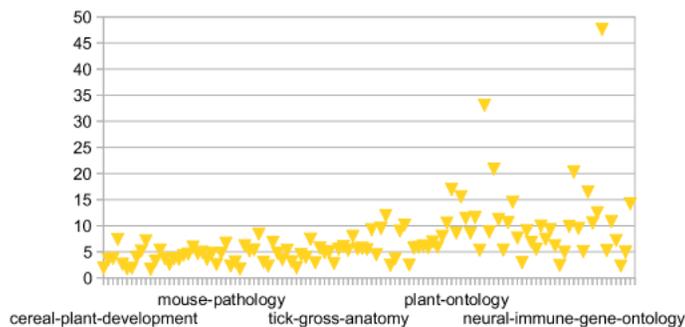
The first set of experiments shows the difference between original and improved module extraction algorithms. As a test we perform an atomic decomposition (improved algorithm) over several well-known ontologies, because it intensively uses the module extraction procedure. The results are presented in Table 1. Here the ontology size is given in the number of axioms, size of the atomic decomposition (AD size) is given in number of atoms.

Table 1. Time and number of locality checks for some ontologies

Ontology	Ont. size, #axioms	AD size, #atoms	Old algorithm		New Algorithm		Ratio
			time, sec	nLoc	time, sec	nLoc	
NCI	85,685	54,332	2,282.0	$17.3 \cdot 10^9$	521.2	$330 \cdot 10^6$	52.4
GO	25,117	25,114	414.0	$2.3 \cdot 10^9$	39.6	$88 \cdot 10^6$	26.1
Galen (Full)	4,979	2,699	4.6	$56.7 \cdot 10^6$	2.9	$5.7 \cdot 10^6$	9.9
Wine	869	5	0.0	$125 \cdot 10^3$	0.0	$56 \cdot 10^3$	2.2

This table shows some general patterns of the performance improvement. The first two ontologies represent the case of a large number of small atoms, where the improved algorithm behaves in the best way. The full version of the Galen ontology is very hard for reasoning. It contains one large atom (about 950 axioms), while all other atoms are rather small. Still, the improved algorithm requires only 10% of the locality checks in the original one. The Wine ontology represents the other end of the spectrum: a few very large atoms. This leads to the smallest improvement of the new algorithm against the original one; however, even in this case it uses 50% operations of the standard algorithm.

Fig. 1. Ratio between the improved and original atomic decomposition algorithms on BioPortal ontologies



The second set of experiments compares two atomic decomposition algorithms on a set of ontologies. We use the OWL API implementation as a refer-

ence, and the FaCT++ implementation as an improved algorithm. The set of test ontologies is a subset of BioPortal ontologies, described in [5].

The results of the tests are shown at Fig. 1. The graph shows the ratio between the time needed to decompose ontologies by the original algorithm and the improved algorithm. While the average ratio is about 7, in the extreme cases the improved algorithm demonstrates 48 times better performance.

6 Conclusions

We propose new improved algorithms of the locality-based module extraction and atomic decomposition of the ontologies. We prove their correctness and compare them with the original algorithms. Provided empirical evaluation results confirm that the proposed algorithms outperformed original ones on a set of real-life ontologies.

We are planning to implement a semantic locality checker and compare results on real-life ontologies. We also are planning to implement *labelled atomic decomposition* [5], which could be useful for the fast module extraction.

References

1. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: atomic decomposition. In: Proc. of IJCAI. pp. 2232–2237 (2011)
2. Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research 31(1), 273–318 (2008)
3. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal properties of modularisation. Modular Ontologies pp. 25–66 (2009)
4. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. Automated Reasoning pp. 292–297 (2006)
5. Vescovo, C.D., Gessler, D., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Winget, A.: Decomposition and modular structure of bioportal ontologies. In: International Semantic Web Conference (1). pp. 130–145 (2011)

Incremental Query Rewriting for OWL 2 QL

Tassos Venetis, Giorgos Stoilos, and Giorgos Stamou

School of Electrical and Computer Engineering
National Technical University of Athens, Greece

1 Introduction

A key application of Description-Logic based ontologies is Ontology-Based Data Access (OBDA) [9]. In such scenarios a TBox is used to describe the schema of the application while answers to conjunctive queries reflect both the schema and the data. Unfortunately, it is well-known that conjunctive query (CQ) answering over expressive Description Logics (DLs) is of very high computational complexity [7, 5].

The need for efficient query answering has motivated the development of (families of) *lightweight* ontology languages, such as the DL-Lite family [3, 2]. Query answering in these languages is usually performed via a technique called *query rewriting*. According to this technique, a query and a DL-Lite ontology are transformed into a union of conjunctive queries (often called a *UCQ rewriting*) such that, the answers of the union of conjunctive queries over the input data and discarding the ontology are precisely the answers of the original query over the data and the ontology.

In the last years a large number of different algorithms and systems for computing rewritings for DL-Lite ontologies has been presented. Examples of such systems are QuOnto [1], Requiem [8], Presto [10], Nyaya [6], and Rapid [4]. Roughly speaking, all systems apply a set of equivalence-preserving transformations over the input query and TBox producing new queries until a fix-point is reached. In several previous approaches [3, 8, 6] this process is largely brute-force, in the sense that the algorithm iterates over the currently computed set of queries, over the atoms of the query and over the TBox axioms, and if some of the rules of the algorithm applies then a new query is generated.

It was shown recently that given a query q , a TBox \mathcal{T} and an atom α of q , a UCQ rewriting for q, \mathcal{T} can be computed by first computing a UCQ rewriting u^- for query $q \setminus \{\alpha\}$ (i.e., q without the atom α) and then ‘extending’ this rewriting with additional information from \mathcal{T} that *only* regards α [11]. Using this idea we present a novel algorithm for computing a UCQ rewriting for queries over DL-Lite_R-TBoxes¹ incrementally. Roughly speaking, given a query q with atoms $\alpha_1, \dots, \alpha_n$ the algorithm first computes UCQ rewritings u_i for ‘special’ queries that contain only a single body atom α_i . Finally, these UCQs are iteratively ‘combined’ until a UCQ rewriting for the input query has been computed.

Compared to several previous approaches our algorithm is significantly guided. At each step all the knowledge of \mathcal{T} that regards a single atom α_i is ‘materialised’

¹ DL-Lite_R is a popular member of the DL-Lite family [3].

into u_i and is used to extend the currently computed UCQ. Our approach also shows that the process of rewriting (at least for DL-Lite) can be largely performed in parallel by ‘decomposing’ q into parts and processing them separately, which to the best of our knowledge, was previously unknown.

Furthermore, to further increase the efficiency of the algorithm, we additionally present a list of optimisations which considerably decrease its computation time. Many of the optimisations are intended to increase the efficiency of our final backward-subsumption (redundancy elimination) algorithm.

Finally, we have implemented the proposed algorithm and optimisations and we have compared them against several available state-of-the-art systems. Our results show that computing a UCQ rewriting incrementally is in the vast majority of cases more efficient than all systems. More precisely, our algorithm requires less time and computes the smallest UCQ rewriting in nearly all ontologies. Interestingly, when compared to the original DL-Lite algorithm [3], which also uses the same technique to compile knowledge from \mathcal{T} , our algorithm is several orders of a magnitude faster, which shows the benefits of the more guided approach.

An extended version of the paper with detailed proofs of correctness can be found online.²

2 Preliminaries

Let \mathbf{C} , \mathbf{R} , and \mathbf{I} be countable, pairwise disjoint sets of *atomic concepts*, *atomic roles*, and *individuals*. A *DL-Lite_R-role* is either an atomic role P or its *inverse* P^- . *DL-Lite_R-concepts* are defined inductively by the grammar $B := A \mid \exists R$, where $A \in \mathbf{C}$ and R is a DL-Lite_R-role. A *DL-Lite_R-TBox* is a finite set of axioms of the form $B_1 \sqsubseteq B_2$ or $B_1 \sqcap B_2 \sqsubseteq \perp$, with $B_{(i)}$ DL-Lite_R-concepts and \perp the *bottom* concept that is empty in all interpretations, or of the form $R_1 \sqsubseteq R_2$ with $R_{(i)}$ DL-Lite_R-roles. An *ABox* is a finite set of assertions of the form $A(c)$ or $P(c, d)$ for $A \in \mathbf{C}$, $P \in \mathbf{R}$ and $c, d \in \mathbf{I}$. A DL-Lite_R-ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ consists of a TBox and an ABox.

A *conjunctive query* (CQ) q is an expression of the form $\{\vec{x} \mid \{\alpha_1, \dots, \alpha_m\}\}$ where $\{\alpha_1, \dots, \alpha_m\}$ is called the *body* of the query with α_i a concept or role atom of the form $A(t)$ or $R(t, t')$ (for t, t' function-free terms and A, R atomic) and $\vec{x} = (x_1, \dots, x_n)$ is a tuple of variables called the *distinguished* (or answer) variables, each appearing in at-least some atom α_i . The remaining variables of q are called *undistinguished*. We use $\text{var}(q)$ to denote all the variables appearing in q and $\text{avar}(q)$ to denote all its distinguished variables. We often abuse notation and use q to refer to the set of its atoms, i.e., $\{\alpha_1, \dots, \alpha_m\}$. Hence, for β an atom and q a CQ, $q \cup \{\beta\}$ denotes a new CQ that consists of the atoms of q plus β and the same distinguished variables as q . For the rest of the paper, and without loss of generality, we will assume that queries are *connected* [5]. Finally, a union of conjunctive queries (UCQ) is a set of CQs.

Given CQs q_1, q_2 with distinguished variables \vec{x} and \vec{y} , respectively, we say that q_2 *subsumes* q_1 , if there exists a substitution θ from the variables of q_2 to the

² <http://image.ece.ntua.gr/~gstoil/main.pdf>

variables of q_1 such that the set $\{\{Q(\vec{y})\} \cup q_2\}_\theta$ is a subset of the set $\{Q(\vec{x})\} \cup q_1$, where Q is a predicate of the same arity as \vec{x} and \vec{y} that does not appear in q_1 or q_2 . Finally, for a UCQ u and CQ q , we say that q is *redundant* in u if another query in u exists that subsumes q ; otherwise it is called *non-redundant*.

For a DL-Lite_R-TBox, a UCQ rewriting u for q, \mathcal{T} can be computed using the *perfect reformulation* algorithm (PerfectRef) [3]. The algorithm applies exhaustively a *reformulation* and a *reduction* step that generate new CQs; the process terminates when no new CQ is generated. More precisely, in the reformulation step the algorithm picks a CQ q , an atom α in the body of the CQ and an axiom I in \mathcal{T} and *applies* the axiom on α replacing it with a new atom. For example, for the query $q_1 = \{x \mid \{R(x, y), A(y)\}\}$ and the axiom $I_1 = \exists R \sqsubseteq A$, applying I_1 on atom $A(y)$ produces the new CQ $q_2 = \{x \mid \{R(x, y), R(z, y)\}\}$, where z is a ‘fresh’ variable. In the reduction step a new CQ is generated by applying to some CQ q the most general unifier (mgu) of two of its atoms. For example, applying reduction on query q_2 above generates query $q_3 := \{x \mid \{R(x, y)\}\}$.

Let $G = \langle U, E \rangle$ be a graph. For $a, b \in U$ we say that b is *reachable* from a , written $a \rightsquigarrow_G b$, if c_0, \dots, c_n with $n \geq 0$ exist where $c_0 = a$, $c_n = b$ and $\langle c_i, c_{i+1} \rangle \in E$ for each $0 \leq i < n$. An element $c \in U$ is called *top* in G if for each $c' \in U$ we have $c \rightsquigarrow_G c'$.

3 Extending Query Rewritings

It has been shown in [11] that given a CQ q , a rewriting u for q, \mathcal{T} and an atom α , a UCQ rewriting for the query $q' = q \cup \{\alpha\}, \mathcal{T}$ can be computed by re-using the previously computed (given) information for q . Roughly speaking, the algorithm computes a UCQ rewriting u_α for a query q_α that consists only of the atom α and then extends the queries in u with atoms of the queries from u_α . The following example illustrates this idea.

Example 1. Consider the following DL-Lite_R-TBox and CQ:

$$\mathcal{T} = \{\text{Professor} \sqsubseteq \exists \text{teaches}, \exists \text{teaches}^- \sqsubseteq \text{Student}\} \quad q = \{x \mid \{\text{teaches}(x, y)\}\}$$

and the UCQ rewriting $u = \{q, q_1\}$ where $q_1 = \{x \mid \{\text{Professor}(x)\}\}$ for q, \mathcal{T} computed using PerfectRef. Assume now, that q is extended in order to retrieve only those individuals that teach students—that is, q is extended to $q' = \{x \mid \{\text{teaches}(x, y), \text{Student}(y)\}\}$. In order to compute a UCQ rewriting for q', \mathcal{T} the algorithm presented in [11] proceeds as follows.

First, it constructs the query $q_\alpha = \{y \mid \{\text{Student}(y)\}\}$ that consists of the single body atom α and its distinguished variables are the common variables between α and q . Then, a UCQ rewriting $u_\alpha = \{q_\alpha, q'_\alpha\}$ for q_α, \mathcal{T} is computed using PerfectRef, where $q'_\alpha = \{y \mid \{\text{teaches}(z, y)\}\}$ for z a fresh variable. Subsequently, the algorithm initialises an empty UCQ u' and iterates over the sets u and u_α constructing and adding new queries to u' as follows:

1. The atoms of q_α are added to q ; hence, query q' is added to u' .

2. The atoms of q'_α are added to q ; hence, query $q'_1 = q \cup \{\text{teaches}(z, y)\}$ is added to u' .
3. The algorithm identifies that the body atom of q'_α can be unified into the body of q ; the result (i.e., CQ q) is added to u' . Additionally, since after this unification CQ q is part of the target UCQ u' all queries that are produced in u due to q also need to be added; hence, query q_1 is also added to u' .
4. No query is generated from q_1 and q_α (or q'_α) since q_1 does not contain all the distinguished variables of q_α (or q'_α), i.e., $\text{avar}(q_\alpha) \not\subseteq \text{var}(q_1)$.

It can be verified that the set $u' = \{q', q'_1, q, q_1\}$ is a UCQ rewriting for q', \mathcal{T} . \diamond

Intuitively, the above approach is possible because the process of rewriting is to a large extent ‘local’ with respect to the atoms of a query. For example, the application of reformulation on some query atom is independent from the other atoms of the query, hence the information from \mathcal{T} that regards α can be materialised and then used to extend the queries in u . The only exception is the reduction step where two different atoms are unified. This step was introduced in [3] because an axiom might only be applicable to a reduction of some query—that is, after reduction the reformulation procedure can continue. To tackle these cases the algorithm in [11] checks whether a query from u_α can be ‘absorbed’ (‘merged’) into a query q_i from u . Note, however, that the algorithm does not apply exhaustively all possible unifications as done in the original reduction step. In contrast, it unifies a query q_α into a query q in such a way that the queries that are (possibly) produced in u due to q can still be produced. This is similar to the *factorisation* optimisation [6]. Our algorithm uses the following function.

Function mergeCQs: Let q, q' be two queries. Then, function $\text{mergeCQs}(q', q)$ returns a substitution σ defined as follows: (i) if there exists $\alpha \in q' \cap q$, then σ is the identity substitution; (ii) if there exist $R(z, y) \in q', R(x, y) \in q$ or $R(y, z) \in q', R(y, x) \in q$ and x, y, z are pair-wise different, then $\sigma = \{z \mapsto x\}$; otherwise, $\sigma = \emptyset$.

In Example 1, for q'_α and q we have $\text{mergeCQs}(q'_\alpha, q) = \{z \mapsto x\}$, hence q as well as all queries that are produced in u due to q (i.e., q_1) are added to the result. To accomplish the latter, however, the algorithm needs to be aware of the dependencies of the queries in the given (pre-computed) UCQ u . To capture this information, instead of a UCQ, the algorithm accepts as input a graph \mathcal{G} of queries which encodes the dependencies between the queries in u .

Definition 1. Let q be a CQ and let \mathcal{T} be a DL-Lite_R-TBox. A rewriting graph for q, \mathcal{T} is a directed graph $\mathcal{G} = \langle u, \mathcal{H}, m \rangle$, where u is a UCQ rewriting for q, \mathcal{T} , \mathcal{H} is a binary relation over u , and each node $q_i \in u$ is labelled with a substitution $m(q_i)$. Moreover, \mathcal{G} satisfies the following properties: (i) If $\langle q_1, q_2 \rangle \in \mathcal{H}$, then q_2 is produced from q_1 by the application of a reformulation or reduction step, and (ii) for each $\langle q_1, q_2 \rangle \in \mathcal{H}$ if q_2 is produced by a reformulation step, then $m(q_2) = m(q_1)$, while if it is produced by a reduction step with σ the mgu, then $m(q_2) = m(q_1) \circ \sigma$.

Algorithm 1 IncrementalRew(q, \mathcal{T})

input: A CQ q and a DL-Lite_R-TBox \mathcal{T} .

- 1: Let S be the set of body atoms in q
- 2: Remove an atom α from S s.t. $\text{var}(\alpha) \cap \text{avar}(q) \neq \emptyset$
- 3: Set $av := \text{var}(\alpha) \cap \text{avar}(q)$ and $c_v := \text{var}(\alpha)$
- 4: $\mathcal{G}_i := \text{ex-PerfectRef}(\{av \mid \{\alpha\}\}, \mathcal{T})$
- 5: **while** $S \neq \emptyset$ **do**
- 6: Remove an atom α' from S s.t. $\text{var}(\alpha') \cap c_v \neq \emptyset$
- 7: $qv := c_v \cap \text{var}(\alpha')$
- 8: $av := av \cup (\text{var}(\alpha') \cap \text{avar}(q))$
- 9: $\mathcal{G}_\alpha := \text{ex-PerfectRef}(\{qv \mid \{\alpha'\}\}, \mathcal{T})$
- 10: $\mathcal{G}' := \langle u', \mathcal{H}', m' \rangle$ for an empty UCQ u' , binary relation \mathcal{H}' and mapping m'
- 11: Let q_i be a top CQ in \mathcal{G}_i and q_α a top CQ in \mathcal{G}_α
- 12: **joinGraphs**($q_i, q_\alpha, \mathcal{G}', \mathcal{G}_i, \mathcal{G}_\alpha, av, qv$)
- 13: $c_v := c_v \cup \text{var}(\alpha')$
- 14: $\mathcal{G}_i := \mathcal{G}'$
- 15: **return** removeRedundant(\mathcal{G})

A rewriting graph for a query q over a DL-Lite_R-TBox can be easily computed by a straightforward extension of the PerfectRef algorithm, which we call ex-PerfectRef. The details of the algorithm have been presented in [11].

4 An Incremental Query Rewriting Algorithm

The previous results show that a rewriting for a given (fixed) query over some TBox can be computed incrementally by considering one of its atoms at a time. For example, for query $q' = \{x \mid \{\text{teaches}(x, y), \text{Student}(y)\}\}$ of Example 1 we can first select atom $\alpha_1 := \text{teaches}(x, y)$ compute a UCQ rewriting u_{α_1} for $q_{\alpha_1} = \{x \mid \{\text{teaches}(x, y)\}\}$ (which consists of the set $\{q, q_1\}$ of the example) and then pick the last atom, compute a UCQ u_{α_2} for $q_{\alpha_2} := \{y \mid \{\text{Student}(y)\}\}$ and finally extend u_{α_1} with atoms of queries from u_{α_2} as shown in Example 1. In general, given a (fixed) query one can pick one of its atoms, compute a rewriting (graph) for it, and then iteratively add the rest of its atoms by extending the previously computed rewriting. When all the atoms have been processed a UCQ rewriting for the given query would have been computed. In contrast to our previous work [11], at each step this algorithm should produce a rewriting graph out of the input rewriting graph instead of a UCQ in order to be able to iteratively process all the atoms.

This idea is illustrated in Algorithm 1. The algorithm first selects some atom α such that some of its variables appear as distinguished variables in q (line 2) and computes a rewriting graph \mathcal{G}_i for the query $\{\text{var}(\alpha) \cap \text{avar}(q) \mid \{\alpha\}\}$ (line 4). Hence, initially a rewriting graph for a query that contains only atom α of q , variables $c_v := \text{var}(\alpha)$ of q and distinguished variables $av := \text{var}(\alpha) \cap \text{avar}(q)$ of q have been computed. Then, the algorithm selects one-by-one the remaining atoms and extends the previously computed rewriting graph (lines 5–14). More

Algorithm 2 $\text{joinGraphs}(q_h, q_\alpha, \mathcal{G}', \mathcal{G}, \mathcal{G}_\alpha, av, jv)$

input: Rewriting graphs $\mathcal{G}' = \langle u', \mathcal{H}', m' \rangle$, $\mathcal{G} = \langle u, \mathcal{H}, m \rangle$ and $\mathcal{G}_\alpha = \langle u_\alpha, \mathcal{H}_\alpha, m_\alpha \rangle$ and two sets of variables jv and av .

- 1: $\kappa := m(q_h)$
- 2: **if** $\text{canBeJoined}(q_h, \kappa, jv)$ **then**
- 3: Create CQ $q_c := \{av \mid q_h \cup (q_\alpha)_\kappa\}$, set $m'(q_c) := \kappa$ and add q_c to u'
- 4: Set $\sigma := \text{mergeCQs}(q_\alpha, q_h)$
- 5: **if** $\sigma \neq \emptyset$ **then**
- 6: Add $\langle q_c, (q_h)_\sigma \rangle$ to \mathcal{G}'
- 7: **for all** q' s.t. $q_h \rightsquigarrow_{\mathcal{G}} q'$ **do**
- 8: Set $m'(\{av \mid q'\}_\sigma) := \kappa \circ \sigma$
- 9: **for all** $\langle q', q'' \rangle \in \mathcal{G}$ **do** Add $\langle \{av \mid q'\}_\sigma, \{av \mid q''\}_\sigma \rangle$ to \mathcal{G}'
- 10: **for all** $\langle q_\alpha, q' \rangle \in \mathcal{G}_\alpha$ **do**
- 11: Create CQ $q'_c := \{av \mid q_h \cup (q')_\kappa\}$, set $m'(q'_c) := \kappa$ and add $\langle q_c, q'_c \rangle$ to \mathcal{G}'
- 12: $\text{joinGraphs}(q_h, q', \mathcal{G}', \mathcal{G}, \mathcal{G}_\alpha, av, jv)$
- 13: **for all** $\langle q_h, q' \rangle \in \mathcal{G}$ **do**
- 14: **if** $\text{canBeJoined}(q', m(q'), jv)$ **then**
- 15: Create CQ $q'_c := \{av \mid q' \cup (q_\alpha)_\kappa\}$, set $m'(q'_c) := \kappa$ and add $\langle q_c, q'_c \rangle$ to \mathcal{G}'
- 16: $\text{joinGraphs}(q', q_\alpha, \mathcal{G}', \mathcal{G}, \mathcal{G}_\alpha, av, jv)$

precisely, at the beginning of the i -th iteration the algorithm has computed a rewriting graph \mathcal{G}_i for a query q_i that contains i atoms of q , c_v contains the variables of q that appear in q_i , while av the distinguished variables of q that appear in q_i . Hence, it picks another atom α' such that some of its variables also appear in c_v (line 6), it adds the variables of α' that are distinguished in q to av (line 8), it computes a rewriting graph \mathcal{G}_α for the query $\{\text{var}(\alpha') \cap c_v \mid \{\alpha'\}\}$ (line 9) and then, it joins \mathcal{G} with \mathcal{G}_α using function joinGraphs (line 12) storing the result to \mathcal{G}' . Finally, after processing all atoms of the query it uses the well-known redundancy elimination algorithm proposed in [8] to remove the redundant (subsumed) queries (line 15).

Function joinGraphs is shown in Algorithm 2. Intuitively, this algorithm computes the Cartesian product of the two input rewriting graphs (modulo cases where queries should be merged). The intuition is that if $\langle q, q' \rangle \in \mathcal{G}$ (i.e., q' is produced by q) and q_α is a vertex in \mathcal{G}_α , then the same step would also be applicable to query $q \cup q_\alpha$ —that is, $q \cup q_\alpha$ will produce the CQ $q' \cup q_\alpha$ (for-loop in line 13). Similarly, for q a vertex in \mathcal{G} and $\langle q_\alpha, q'_\alpha \rangle \in \mathcal{G}_\alpha$ (for-loop in line 10). In addition the algorithm also checks whether a CQ from \mathcal{G}_α can be merged into some CQ q_h from \mathcal{G} (line 4). If this is the case then the queries that have been produced in \mathcal{G} due to q_h are copied to the new rewriting graph (see lines 7–9).

Note that the graphs can be cyclic but standard graph-traversal algorithms can be used to guarantee termination.

Example 2. Consider the TBox \mathcal{T} and CQ $q' = \{x \mid \{\text{teaches}(x, y), \text{Student}(y)\}\}$ of Example 1. A run of Algorithm 1 is the following:

(1.) First, it selects atom α s.t. $\text{var}(\alpha) \cap \text{avar}(q') \neq \emptyset$ (line 2). The only atom that satisfies this condition is $\alpha = \text{teaches}(x, y)$. Subsequently, algorithm

ex-PerfectRef is executed for $q = \{x \mid \{\text{teaches}(x, y)\}\}$ and \mathcal{T} . This creates the rewriting graph $\mathcal{G}_i = \langle u, \mathcal{H}, m \rangle$, where $u = \{q, q_1\}$ is as defined in Example 1, $\mathcal{H} = \{\langle q, q_1 \rangle\}$ and $m(q) = m(q_1) = \emptyset$ (line 4). At this point $c_v = \{x, y\}$ and $av = \{x\}$.

(2.) Next, the algorithm picks another atom α' of q' s.t. $\text{var}(\alpha') \cap c_v \neq \emptyset$. One such atom is $\alpha' = \text{Student}(y)$. Hence, using again algorithm ex-PerfectRef it computes for $q_\alpha = \{y \mid \{\text{Student}(y)\}\}$ and \mathcal{T} (line 9) the rewriting graph $\mathcal{G}_\alpha = \langle u_\alpha, \mathcal{H}_\alpha, m_\alpha \rangle$, where $u_\alpha = \{q_\alpha, q'_\alpha\}$ is as defined in Example 1, $\mathcal{H} = \{\langle q_\alpha, q'_\alpha \rangle\}$ and $m(q_\alpha) = m(q'_\alpha) = \emptyset$. Subsequently, it calls algorithm joinGraphs with parameters $q, q_\alpha, \mathcal{G}', \mathcal{G}_i$ (as computed in the previous step), \mathcal{G}_α and the variable sets $av = \{x\}$ and $av = \{y\}$ in order for \mathcal{G}' to reflect the new graph. This algorithm proceeds as follows: first, it selects q from \mathcal{G}_i and q_α from \mathcal{G}_α and creates the query $q' = \{x \mid \{\text{teaches}(x, y), \text{Student}(y)\}\}$ (by adding atoms of q_α to q) (line 3). Then, it proceeds to the child of q_α , (i.e., to q'_α) and it creates the CQ $q'_1 = \{x \mid \{\text{teaches}(x, y), \text{teaches}(z, y)\}\}$ (by adding atoms of q'_α to q) (line 11). Moreover, it also adds the relation $\langle q', q'_1 \rangle$ to \mathcal{G}' . Subsequently, a recursive call to joinGraphs is made with first two parameters q and q'_α . In this call, in line 4, mergeCQs(q'_α, q) returns $\{z \mapsto x\}$, hence tuples $\langle q'_1, q \rangle$ and $\langle q, q_1 \rangle$ are added to \mathcal{G}' , and $m(q) = m(q_1) = \{z \mapsto x\}$. (Note that $q_\sigma = q$ and $q_{1\sigma} = q_1$) Then, the algorithm returns from the recursive call and proceeds in line 13 to the child of q (i.e., q_1) but canBeJoined($q_1, m(q_1), av$) returns false for the reasons explained in Example 1 item 4. Hence, the algorithm terminates and we have $\mathcal{G}' = \langle u', \mathcal{H}', m' \rangle$ where $u' = \{q', q'_1, q, q_1\}$ (as defined in Example 1), $\mathcal{H}' = \{\langle q', q'_1 \rangle, \langle q'_1, q \rangle, \langle q, q_1 \rangle\}$ and $m(q') = m(q'_1) = \emptyset, m(q) = m(q_1) = \{z \mapsto x\}$. \diamond

5 Optimisations

5.1 Optimising the Last Iteration

As explained earlier, Algorithm 2 computes the cartesian product between two rewriting graphs. The structure of the computed graph is important while processing the atoms of the query, however, it is not important after processing the *last* atom of the input query. Consequently, in the last iteration, Algorithm 1 can call a simplified version of Algorithm 2 that constructs a set of CQs rather than a rewriting graph. Algorithm 3 depicts the simplified algorithm. Roughly speaking, it is obtained from Algorithm 2 by, removing the for-loop starting in line 13, computing for the last selected atom α a set u_α rather than a rewriting graph \mathcal{G}_α , and adding the computed queries to a UCQ rather than a graph.

5.2 Optimising Redundancy Elimination

In line 15 Algorithm 1 applies the well-known redundancy elimination algorithm from [8]. As it has been shown by several experimental evaluations [8, 4], this method usually does not perform well in practice, because it consists of several loops over the (potentially large) set of computed CQs. In order to improve the

Algorithm 3 OptimisedExtensionStep($\mathcal{G}, u_\alpha, jv, av$)

Input: A rewriting graph $\mathcal{G} = \langle u, \mathcal{H}, m \rangle$, a UCQ u_α , and two sets of variables.

- 1: Initialise a queue Q with a top element in \mathcal{G}
- 2: Initialise a UCQ $U := \emptyset$
- 3: **while** $Q \neq \emptyset$ **do**
- 4: Remove the head q of Q and let $\kappa := m(q)$
- 5: **if** canBeJoined(q, κ, jv) **then**
- 6: Add $\{av \mid q \cup (q_\alpha)_\kappa\}$ to U
- 7: **for all** $q_\alpha \in u_\alpha$ **do**
- 8: $\sigma := \text{mergeCQs}(q_\alpha, q)$
- 9: **if** $\sigma \neq \emptyset$ **then**
- 10: **for all** q' s.t. $q \rightsquigarrow_{\mathcal{G}} q'$ **do** Add $\{av \mid q'\}_\sigma$ to U
- 11: **else** Add each q' such that $\langle q, q' \rangle \in \mathcal{G}$ to Q
- 12: **return** U

performance of this method our algorithm uses the following two approaches. First, it tries to identify queries that, if added to the final rewriting, they are going to be redundant. Clearly, such queries need not be added, hence reducing the size of the set over which algorithm `removeRedundant` would be executed. Secondly, it also tries to identify queries that are going to be non-redundant. Such queries can then be excluded from the final check. Our algorithm identifies such queries as follows.

In the last iteration and before calling Algorithm 3 it executes the standard subsumption checking algorithm over \mathcal{G} and stores all subsumption relations. Note that, the size of \mathcal{G} at this point is expected to be significantly smaller than that of the final UCQ, hence the algorithm should behave well in practice. Then, when executing Algorithm 3 it identifies redundant queries as follows:

- In line 10, it adds a query $\{av \mid q'\}_\sigma$ to U *only if* for q the subsumer of q' (if it exists) $\{av \mid q\}$ is not already in U .
- Let q selected in line 4. If a subsumer q' of q exists such that, either $\{av \mid q'\}$ is already in U , or $q' \subseteq q$, $m(q') = m(q)$, and `canBeJoined`($q', m(q'), jv$) = `true`, then the algorithm ‘skips’ q —that is, it adds each q'' such that $\langle q, q'' \rangle \in \mathcal{G}$ to Q and it continues with the next CQ.

Also, Algorithm 3 is modified to identify non-redundant queries as follows:

- At the beginning it initialises an empty set NR of non-redundant queries.
- In line 10, if $\{av \mid q'\}_\sigma = \{av \mid q'\}$ and q' is non-redundant in u it adds $\{av \mid q'\}_\sigma$ to NR .
- In line 6, it adds $\{av \mid q \cup (q_\alpha)_{m(q)}\}$ to NR if none of the predicates in q_α appear in any CQ in u and if for each $q'_\alpha \in u_\alpha$ we have `mergeCQs`(q'_α, q) = \emptyset .
- Finally, it returns both the UCQ U and the set NR .

Subsequently, the returned set NR is used by method `removeRedundant` as follows: All queries that are in the set NR are excluded from redundancy checking.

Table 1: Comparison between PerfectRef, Nyaya, Rapid, and versions of IQAROS

O	Q	Size of UCQ					UCQ Computation Time						Overall Rewriting Time						
		PR	Nyaya	Rapid	Inc ₁	Inc ₂	Inc ₃	PR	Nyaya	Rapid	Inc ₁	Inc ₂	Inc ₃	PR	Nyaya	Rapid	Inc ₁	Inc ₂	Inc ₃
P5	1	6	6	6	6	6	1	14	7	1	1	2	1	14	7	1	2	2	
	2	11	10	10	10	10	15	128	14	7	3	3	17	128	15	8	4	4	
	3	22	13	13	13	13	256	726	22	76	19	19	261	726	23	78	21	21	
	4	45	15	15	15	15	1828	1889	33	288	173	166	1830	1889	36	291	178	169	
	5	90	16	16	16	16	32255	16062	75	838	306	308	32270	16062	77	841	310	310	
P5X	1	14	14	14	14	14	0	12	10	0	0	1	1	12	10	0	1	1	
	2	86	66	25	81	25	2	130	23	2	3	1	13	170	26	6	4	3	
	3	530	374	127	413	133	74	36	540	92	24	6	12	150	1415	135	95	19	36
	4	3476	2475	636	2070	670	393	656	1672	343	187	46	122	5876	3842	1181	313	283	445
	5	23744	17584	3180	10352	3352	2057	41454	15095	2061	828	214	371	326400	142580	5252	2817	1078	1233
S	1	6	6	6	6	6	0	15	6	0	0	0	0	15	6	0	0	0	
	2	202	3	2	204	12	2	12	11	9	12	4	3	34	12	9	12	4	3
	3	1005	7	4	864	96	4	190	46	14	60	8	7	677	48	14	65	9	7
	4	1548	5	4	1428	84	4	254	34	14	104	9	6	889	34	14	116	10	7
	5	8693	13	8	6048	672	8	8216	159	36	1018	227	93	54252	163	37	1146	236	93
U	1	2	2	2	2	2	0	25	9	1	1	2	1	25	9	1	1	2	
	2	189	1	1	190	5	1	24	7	19	12	3	4	32	7	19	12	3	4
	3	296	4	4	300	20	4	112	172	13	77	5	7	144	172	14	79	5	7
	4	1763	2	2	1688	45	2	826	15	17	253	8	10	1500	15	17	258	8	10
	5	3418	11	10	3375	90	10	2680	107	18	527	17	20	5083	108	19	582	18	20
UX	1	5	5	5	5	5	0	24	11	1	1	2	0	24	11	1	2	2	
	2	286	1	1	287	7	1	14	6	13	10	4	4	31	6	13	11	4	4
	3	1248	12	12	1260	84	12	118	166	20	80	10	11	534	166	21	104	21	11
	4	5385	5	5	5137	129	5	829	15	17	201	11	13	6354	15	17	243	15	13
	5	9220	26	25	8955	225	25	2625	115	26	427	31	53	19622	120	30	593	67	53
A	1	402	248	27	357	77	77	24	1231	18	17	5	10	55	1304	18	18	11	14
	2	103	93	54	103	54	54	124	4928	43	39	12	16	127	4967	45	39	43	17
	3	104	105	104	104	104	104	656	35451	97	173	103	106	661	35491	97	177	328	107
	4	492	455	333	471	320	320	1237	17121	170	170	58	52	1297	17511	208	197	130	55
	5	624	-	624	624	624	624	355571	-	383	3412	258	620	355872	-	384	3667	491	637
AX	1	783	556	41	794	431	431	30	1282	26	18	4	10	135	1649	26	24	8	13
	2	1812	1738	1546	1812	1653	1545	141	4493	649	57	26	30	892	5588	1191	752	772	92
	3	4763	4742	4466	4763	4466	4466	707	34032	1694	186	48	125	8244	51352	2225	10018	8006	491
	4	7251	6565	4497	7229	6639	4479	1282	16569	1247	192	37	45	12782	36460	2785	4891	3579	304
	5	78885	-	32956	78885	74025	32960	319681	-	3810	4361	665	1276	-	-	60006	-	-	26770

Note that, some of the conditions above might seem rather strict. However, as shown by our experimental evaluation these are usually satisfied in practice and they can indeed be very effective. Moreover, their implementation overhead can be noticeable in some cases, however, their benefits in several difficult scenarios greatly outperforms it.

6 Evaluation

We have implemented Algorithms 1–3 in a prototype tool called IQAROS³ and have compared it against PerfectRef [3], Nyaya [6], Requiem [8], and Rapid [4].⁴ Regarding IQAROS we included three versions; the first one (Inc₁) implements Algorithms 1 and 2 without any optimisations; the second one (Inc₂) uses Algorithm 3 instead of Algorithm 2 when it adds the last atom of the query; the

³ <http://code.google.com/p/iqaros/>

⁴ We were not able to obtain Presto as it is not publicly available. We also do not present Requiem due to space limitations and since Rapid outperforms it.

third one (Inc_3) refines Inc_2 by also implementing the various optimisations detailed in the previous section. For the evaluation we used the relatively standard framework proposed in [8], however, we did not include results for ontologies V and P1 since they are rather trivial for all systems. Experiments were conducted on a MacBook Pro with a 2.66GHz processor and 4GB of RAM, with a time-out of 600 seconds.

Table 1 presents the results for each system, where the columns annotated as “Size of UCQ” present the size of the computed UCQ before the final redundancy elimination (after redundancy elimination all systems return the same UCQ, as the ones reported in [8]), while the rest present the computation time before and after redundancy elimination (measured in milliseconds).

First we compare the different versions of IQAROS. We observe that the size of the computed UCQs decreases from Inc_1 to Inc_3 . The difference between Inc_1 and Inc_2 is justified by the fact that the latter uses the simpler algorithm (Algorithm 3) which does not compute the Cartesian product between the graphs. The benefits of using this algorithm are also reflected in the computation times of Inc_2 compared to Inc_1 . Inc_3 computes the smallest UCQ of the three versions due to its techniques for eliminating redundant queries. Regarding execution time Inc_3 performs similarly to Inc_2 and sometimes slightly worse, due to the overhead of implementing the various optimisations. However, when considering the total time the benefits of the optimisations become apparent. Inc_3 is significantly faster in ontologies A and AX and is actually the only configuration of IQAROS that can process query 5 in AX in only 27 seconds. This is heavily due to the optimisation of tracking non-redundant queries.

Compared to PerfectRef, and Nyaya, all versions of IQAROS (even Inc_1) are much faster, in some cases even for several orders of a magnitude. Moreover, Inc_2 and Inc_3 compute significantly smaller UCQs. Since in their core all these systems are based on the same approach for materialising knowledge from \mathcal{T} , we concluded that this improvement is due to the incremental rewriting strategy that provides a much more guided and localised strategy compared to the blind brute-force application of the reformulation and reduction steps. Also Nyaya supports n -ary predicates and its factorisation step is significantly more involved than our merge function.

Compared to Rapid, Inc_3 (the fastest of the three configurations) computes similarly small UCQs with some small exceptions (either against or in favor) in queries 3–5 in ontology P5X, in queries 1 and 3 in ontology A and in queries 1, 2, 4 and 5 in ontology AX. Moreover, Rapid is notably faster⁵ only in queries 4 and 5 in P5 and 5 in S and A. However, even in these cases the difference between the systems is rather marginal as it never exceeds 253 milliseconds. In all the other cases Inc_3 is faster with most notable cases queries 4 and 5 in P5X and 2–5 in AX. Moreover, we can also see that redundancy elimination algorithm of Inc_3 is much more efficient than that of Rapid with again notable case query 5 in ontology AX. Once more, this is justified by the optimisations used in Inc_3 .

⁵ We consider a system to be ‘notably faster’ if it is faster for more than 20 milliseconds.

7 Conclusion

In the current paper we presented a novel algorithm for query rewriting over DL-Lite_R ontologies. The algorithm is based on a novel approach that processes each atom separately and then combines the results to compute a final UCQ rewriting. It is significantly guided and our experimental evaluation showed that it is generally faster than all available systems known to us.

We feel that our techniques have several important practical and theoretical consequences and give opportunities for future work. First, we strongly feel that this approach can be used in other First-Order rewritable languages, like *Linear-Datalog*[±] [6], and there is strong evidence that the resulting system would exhibit good performance. Even in non-First-Order rewritable languages one could perhaps still exploit parts of this technique to increase the efficiency of the rewriting algorithms. Moreover, our results show that the rewriting process (at-least for DL-Lite) can largely be performed in parallel and such techniques can be further investigated.

Acknowledgments Work supported by project EUscreen (ECP-2008-DILI-518002) within EU's eContentplus Programme. Giorgos Stoilos is supported by a Marie Curie FP7-Reintegration-Grants within European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement 303914.

References

1. Acciarri, A., Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: Quonto: Querying ontologies. In: Proc. of AAAI-05. (2005)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: The DL-Lite family and relations. *Journal of Artificial Intelligence Research* 36, 1–69 (2009)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning* 39(3), 385–429 (2007)
4. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting in OWL 2 QL. In: Proc. of CADE-23 (2011)
5. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic *SHIQ*. In: Proc. of IJCAI 2007 (2007)
6. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of ICDE 2011 (2011)
7. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Proc. of IJCAR 08. pp. 179–193 (2008)
8. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient Query Answering for OWL 2. In: Proc. of ISWC-09 (2009)
9. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. *Journal on Data Semantics* X, 133–173 (2008)
10. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proc. of KR-10 (2010)
11. Venetis, T., Stoilos, G., Stamou, G.: Query rewriting under query extensions for OWL 2 QL ontologies. In: Proc. of SSWS-11, Bonn, Germany (2011)

Absorption for ABoxes

Jiewen Wu, Alexander Hudek, David Toman, and Grant Weddell

Cheriton School of Computer Science
University of Waterloo, Canada
{j55wu, akhudek, david, gweddell}@uwaterloo.ca

Abstract. We present a novel method of evaluating instance queries over description logic knowledge bases that derives from binary absorption. The method is designed to work well for large ABoxes and where the TBox is not necessarily Horn, e.g., where background knowledge requires the use of disjunction and negation. The method significantly improves the performance of instance checking, and particularly so in cases where a large number of concrete feature values are included. We also report on the results of a preliminary experimental evaluation that validates the efficacy of the method.

1 Introduction

Two of the basic reasoning tasks over a DL knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ are to determine if \mathcal{K} is consistent, and so-called *instance checking*: to determine if a given concept assertion $a : C$ (stating that individual a occurring in \mathcal{A} belongs to concept C) is a logical consequence of a *consistent* \mathcal{K} , written $\mathcal{K} \models a : C$.

Usually, these tasks are combined in the sense that the latter is assumed to include the former. However, we believe that typical workloads for a reasoning service will include far more instance checking tasks than knowledge base consistency tasks. The resulting “separation of concerns” can therefore enable technology that is far more efficient for such workloads, particularly so in the case of non-Horn DL TBoxes \mathcal{T} that preclude the possibility of computing so-called canonical ABoxes \mathcal{A}' from \mathcal{A} (e.g., when disjunction is used in \mathcal{T}). We contribute to this development by introducing a novel adaptation of binary absorption for DL knowledge bases and demonstrate that the technique is efficacious for workloads that contain many thousands of instance checking tasks.

To date, work on absorption has focused on the *concept satisfaction* problem, a simple case of the instance checking problem for knowledge bases with an ABox consisting of a single assertion $a : \top$. Indeed, it has been known for some time in this case that *lazy unfolding* is an important optimization technique in model building algorithms for satisfiability [1]. It is also imperative for a large TBox to be manipulated by an *absorption generation* procedure to maximize the benefits of lazy unfolding in such algorithms, thereby reducing the combinatorial effects of disjunction in underlying tableaux procedures [3].

To consider performance issues for instance checking in the context of absorption, we first consider how one can map instance checking problems to concept satisfaction problems in which consistency is assumed, and then revisit absorption in this new setting. In particular, we present an absorption generation procedure that is an adaptation of an earlier procedure reported at the description

logics workshop [6]. This earlier procedure was called *binary absorption* and was itself a generalization of the absorption theory and algorithms developed by Horrocks and Tobies [4, 5]. The generalization makes it possible for lazy unfolding to be used for parts of terminologies not handled by earlier absorption algorithms and theory.

Binary absorption combines two key ideas. The first makes it possible to avoid *internalizing* (at least some of the) terminological axioms of the form $(A_1 \sqcap A_2) \sqsubseteq C$, where the A_i denote *primitive concepts* and C a general concept. The second is an idea relating to *role absorptions* developed by Tsarkov and Horrocks [13]. To illustrate, binary absorption makes it possible to completely absorb the inclusion dependency

$$A_1 \sqcap (\exists R_1^- . A_2) \sqcap (\exists R_2 . (A_3 \sqcup A_4)) \sqsubseteq A_5.$$

In this case, the absorption would consist of a set of dependencies with a single primitive concept on the left-hand-side

$$\{A_2 \sqsubseteq \forall R_1 . A_6, A_3 \sqsubseteq A_7, A_4 \sqsubseteq A_7, A_7 \sqsubseteq \forall R_2^- . A_8\}$$

and a second set of dependencies with a conjunction of two primitive concepts on the left-hand-side

$$\{(A_1 \sqcap A_6) \sqsubseteq A_9, (A_9 \sqcap A_8) \sqsubseteq A_5\},$$

in which A_6, A_7, A_8 and A_9 are fresh atomic concepts introduced by the binary absorption procedure. (Hereon, we refer to an instance of the latter set as a *binary inclusion dependency*.) A key insight and contribution of this paper is that it is not necessary for *both* concepts occurring in the left-hand-side of such a dependency to be atomic. In particular, we show that binary absorption raises the possibility of reducing assertion membership problems to concept satisfaction problems via the introduction of nominals in such dependencies, *but without suffering the consequent overhead that doing so would almost certainly entail without binary absorption*.

Note that there are other reasons that binary absorption is useful, beyond the well-documented advantages of reducing the need for internalization of general terminological axioms. In particular, it works very well for the parts of a terminology that are Horn-like, as illustrated by the above example.

Our contributions are as follows:

1. We introduce the notion of role and concrete feature guards in the context of a knowledge base for the DL dialect $\mathcal{ALCIQ}(\mathbf{D})$. In particular, we show how instance checking tasks in this dialect can map to concept satisfaction problems in the dialect $\mathcal{ALCIOQ}(\mathbf{D})$, but where binary absorption in combination with guards can usefully avoid reasoning about irrelevant ABox individuals and concrete facts with the assumption of knowledge base consistency.
2. We propose a generalization of binary absorption. In particular, we now allow nominals in place of one of the two left-hand-side concepts in an absorbed binary inclusion dependency.
3. We report on the results of an experimental evaluation that validates the efficacy of the proposed optimization.

After some preliminary definitions, these contributions are the subject of successive sections, and are followed in turn by our summary comments. Finally, note that a short earlier version of this paper is being presented simultaneously in a poster session of KR 2012 [14].

2 Preliminaries

We consider instance checking problems in the context of knowledge bases expressed in terms of the DL dialect $\mathcal{ALC}\mathcal{I}\mathcal{O}\mathcal{Q}(\mathbf{D})$. However, such problems will be mapped to concept satisfaction problems in the more general dialect $\mathcal{ALC}\mathcal{I}\mathcal{O}\mathcal{Q}(\mathbf{D})$.

Definition 1 (Description Logic $\mathcal{ALC}\mathcal{I}\mathcal{O}\mathcal{Q}(\mathbf{D})$).

$\mathcal{ALC}\mathcal{I}\mathcal{O}\mathcal{Q}(\mathbf{D})$ is a DL dialect based on disjoint infinite sets of atomic concepts NC, atomic roles NR, concrete features NF and nominals NI. Also, if $A \in \text{NC}$, $R \in \text{NR}$, $a \in \text{NI}$, $f, g \in \text{NF}$, n is a non-negative integer and C_1 and C_2 are concept descriptions, then A , $\neg C_1$, $C_1 \sqcap C_2$, $C_1 \sqcup C_2$, \top , \perp , $\exists R.C_1$, $\forall R.C_1$, $\exists R^-.C_1$, $\forall R^-.C_1$, $\{a\}$, $\exists^{\leq n} R.C_1$, $\exists^{\geq n} R.C_1$, $\exists^{\leq n} R^-.C_1$, $\exists^{\geq n} R^-.C_1$, $f < g$ and $f = k$, where k is a finite string, are also concept descriptions.

An interpretation \mathcal{I} is a pair $\mathcal{I} = (\Delta^{\mathcal{I}} \uplus \mathbf{D}^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, $\mathbf{D}^{\mathcal{I}}$ a disjoint concrete domain of finite strings, and $\cdot^{\mathcal{I}}$ is a function mapping each feature f to a total function $f^{\mathcal{I}} : \Delta \rightarrow \mathbf{D}$, the “=” symbol to the equality relation over \mathbf{D} , the “<” symbol to the binary relation for an alphabetic ordering of \mathbf{D} , a finite string k to itself, NC to subsets of $\Delta^{\mathcal{I}}$, NR to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and NI to elements of $\Delta^{\mathcal{I}}$. The interpretation is extended to compound descriptions in the standard way.

Concrete domain concepts such as $f < k$ are considered to be a shorthand for $(f < g) \sqcap (g = k)$, which, together with $f = k$, may be generalized as $(t_1 \text{ op } t_2)$, where t_1 and t_2 refer to either a concrete feature or a finite string, and $\text{op} \in \{<, =\}$.

Definition 2 (TBox, ABox, and KB Satisfiability).

A TBox \mathcal{T} is a finite set of axioms of the form $C_1 \sqsubseteq C_2$ or $C_1 \doteq C_2$. A TBox \mathcal{T} is called primitive iff it consists entirely of axioms of the form $A \doteq C$ with $A \in \text{NC}$, each $A \in \text{NC}$ appears in at most one left hand side of an axiom, and \mathcal{T} is acyclic. $A \in \text{NC}$ is defined in \mathcal{T} if \mathcal{T} contains $A \sqsubseteq C$ or $A \doteq C$. An ABox \mathcal{A} is a finite set of assertions of the form $a : A$, $a : (f \text{ op } k)$ and $R(a, b)$.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}\mathcal{I}\mathcal{O}\mathcal{Q}(\mathbf{D})$ knowledge base (KB). An interpretation \mathcal{I} is a model of \mathcal{K} , written $\mathcal{I} \models \mathcal{K}$, iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ holds for each $C_1 \sqsubseteq C_2 \in \mathcal{T}$, $C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$ holds for each $C_1 \doteq C_2 \in \mathcal{T}$, $a^{\mathcal{I}} \in A^{\mathcal{I}}$ for $a : A \in \mathcal{A}$, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, and $f^{\mathcal{I}}(a^{\mathcal{I}}) \text{ op } k$ for $a : (f \text{ op } k) \in \mathcal{A}$. A concept C is satisfiable with respect to a knowledge base \mathcal{K} iff there is an \mathcal{I} such that $\mathcal{I} \models \mathcal{K}$ and such that $C^{\mathcal{I}} \neq \emptyset$.

3 On Absorbing an ABox

The absorption proceeds in two steps: first *guards* that allow us to prune the exploration of the ABox during reasoning are added to the ABox assertions (in turn converted into TBox axioms about nominals) and then the resulting TBox is processed by an extended *binary absorption* algorithm.

3.1 Mapping Instance Checking to Subsumption Testing

In this section we convert an $\mathcal{ALCIQ}(\mathbf{D})$ knowledge base \mathcal{K} to a TBox by representing individuals in \mathcal{K} 's ABox by nominals (i.e., in a *controlled* fragment of $\mathcal{ALCIQ}(\mathbf{D})$):

Definition 3 (ABox Conversion). *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base. We define a TBox $\mathcal{T}_{\mathcal{A}}$ for the ABox of \mathcal{K} :*

$$\begin{aligned} \mathcal{T}_{\mathcal{A}} = & \{ \{a\} \sqcap \text{Def}_a \sqsubseteq A \mid a : A \in \mathcal{A} \} \\ & \cup \{ \{a\} \sqcap \text{Def}_f \sqsubseteq (f \text{ op } k) \mid a : (f \text{ op } k) \in \mathcal{A} \} \\ & \cup \{ \{a\} \sqcap \text{Def}_R \sqsubseteq \exists R.(\{b\} \sqcap \text{Def}_b), \{a\} \sqcap \text{Def}_a \sqsubseteq \exists R.\top, \\ & \quad \{b\} \sqcap \text{Def}_{R^-} \sqsubseteq \exists R^-.(\{a\} \sqcap \text{Def}_a), \{b\} \sqcap \text{Def}_b \sqsubseteq \exists R^-. \top \mid R(a, b) \in \mathcal{A} \} \end{aligned}$$

Note that all the axioms resulting from ABox assertions are *guarded* by auxiliary primitive concepts of the form Def_a , Def_R , and Def_f . Intuitively, these concepts, when coupled with an appropriate absorption allow a reasoner to *ignore* parts of the original ABox: all the constants for which Def_a is not *set*, yielding considerable performance gains. For this idea to work we need to require (without loss of generality) that the TBox of \mathcal{K} only uses qualified *at-most* number restrictions of the form $A \sqsubseteq \exists^{\leq n} R.B$ where A and B are atomic concepts or their negations. Note that subsumptions of the form $\exists^{\geq n} R.A \sqsubseteq B$ are also considered to be at-most number restrictions and have to be equivalently rewritten in the above form and that nested restrictions must be unnested. It is easy to see that every $\mathcal{ALCIQ}(\mathbf{D})$ TBox can be transformed to an equi-satisfiable TBox that satisfies this restriction by introducing new auxiliary concept names. Then we add the following assertions that manipulate the guards:

Definition 4 (TBox Conversion). *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base. We define a TBox $\mathcal{T}_{\mathcal{T}}$ for the ABox of \mathcal{K} as follows:*

$$\begin{aligned} \mathcal{T}_{\mathcal{T}} = & \{ A \sqsubseteq \text{Def}_R, B \sqsubseteq \text{Def}_{R^-} \mid A \sqsubseteq \exists^{\leq n} R.B \in \mathcal{T} \} \\ & \cup \{ (t_1 \text{ op } t_2) \sqsubseteq \text{Def}_f \mid f \text{ appears in } t_1 \text{ or in } t_2, (t_1 \text{ op } t_2) \text{ appears in } \mathcal{T} \}. \end{aligned}$$

In the following we use $\mathcal{T}_{\mathcal{K}}$ for $\mathcal{T} \cup \mathcal{T}_{\mathcal{T}} \cup \mathcal{T}_{\mathcal{A}}$.

Theorem 1. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a consistent knowledge base. Then*

$$\mathcal{K} \models a : C \text{ if and only if } \mathcal{T}_{\mathcal{K}} \models \{a\} \sqcap D \sqsubseteq C,$$

where $D = \text{Def}_a \sqcap (\prod_{f \text{ appears in } C} \text{Def}_f)$.

Proof. Assume that there is an interpretation I_0 that satisfies $\mathcal{T}_{\mathcal{K}}$ such that $(\{a\})^{I_0} \subseteq (D)^{I_0}$ but $(\{a\})^{I_0} \cap (C)^{I_0} = \emptyset$ and an interpretation I_1 that satisfies \mathcal{K} in which *all at-least restrictions are fulfilled by anonymous objects*. Hence, we do not need to consider at-least restrictions, no matter how expressed, in the construction below. Without loss of generality, we assume both I_0 and I_1 are tree-shaped outside of the ABox (converted ABox). We construct an interpretation J for $\mathcal{K} \cup \{a : \neg C\}$ as follows: Let Γ^{I_0} be the set of objects $o \in \Delta^{I_0}$ such that either $o \in (\{a\})^{I_0}$ and $(\{a\})^{I_0} \subseteq (\text{Def}_a)^{I_0}$ or o is an anonymous object in Δ^{I_0} rooted by such an object. Similarly let Γ^{I_1} be the set of objects $o \in \Delta^{I_1}$ such that either $o \in (\{a\})^{I_1}$ and $(\{a\})^{I_1} \cap (\text{Def}_a)^{I_1} = \emptyset$ or o is an anonymous object in Δ^{I_1} rooted by such an object. We set

1. $\Delta^J = \Gamma^{I_0} \cup \Gamma^{I_1}$;
2. $(a)^J \in (\{a\})^{I_0}$ for $(a)^J \in \Gamma^{I_0}$ and $(a)^J = (a)^{I_1}$ for $(a)^J \in \Gamma^{I_1}$;
3. $o \in A^J$ if $o \in A^{I_0}$ and $o \in \Gamma^{I_0}$ or if $o \in A^{I_1}$ and $o \in \Gamma^{I_1}$ for an atomic concept A (similarly for concrete domain concepts of the form $(t_1 \text{ op } t_2)$);
4. $(o_1, o_2) \in (R)^J$ if
 - (a) $(o_1, o_2) \in R^{I_0}$ and $o_1, o_2 \in \Gamma^{I_0}$, or $(o_1, o_2) \in R^{I_1}$ and $o_1, o_2 \in \Gamma^{I_1}$; or
 - (b) $o_1 \in (\{a\})^{I_0} \cap (\text{Def}_a)^{I_0}$, $o_2 \in (\{b\})^{I_1}$ and $R(a, b) \in \mathcal{A}$ (or vice versa).

We claim that $(\{a\})^J \cap (C)^J = \emptyset$ (trivially) and $J \models \mathcal{K}$: to show the latter part we only need to consider those R edges of the form covered by the last case (4b): the edges that *cross* between the two interpretations, i.e., when $o_1 \in (\{a\})^{I_0}$, $o_2 \in (\{b\})^{I_1}$ and $R(a, b) \in \mathcal{A}$. Now consider an inclusion dependency expressing an *at-most* restriction $A \sqsubseteq \exists^{\leq n} R.B \in \mathcal{T}$. We can conclude that $o_1 \notin (A)^{I_0}$ as otherwise $o_1 \in (\text{Def}_R)^{I_0}$ by Definition 4 and thus $o_2 \in (\text{Def}_B)^{I_0}$ by Definition 3 which contradicts our assumption that $(\{b\})^{I_0} \cap (\text{Def}_B)^{I_0} = \emptyset$. Hence the inclusion dependency is satisfied vacuously. The remaining edges, case (4a), satisfy all dependencies in \mathcal{K} as the remainder of the interpretation J is copied from one of the two interpretations that satisfy \mathcal{K} . Hence all inclusion dependencies in \mathcal{K} are satisfied by J .

The other direction of the proof follows by observing that if $\mathcal{K} \cup \{a : \neg C\}$ is satisfiable then the satisfying interpretation I can be extended to $(\text{Def}_a)^I = (\text{Def}_f)^I = (\text{Def}_R)^I = \Delta^I$ and $(\{a\})^I = \{a^I\}$ for all individuals a , concrete features f , and roles R . This extended interpretation then satisfies $\mathcal{T}_{\mathcal{K}}$ and $(\{a\})^I \subseteq (D)^I \cap (\neg C)^I$. \square

Also, in $\mathcal{ALCI}(\mathbf{D})$ we do not need to rely explicitly on the unique name assumption (UNA: the logic on its own cannot equate constants). However, we could allow explicit equalities and inequalities to the ABox and then preprocess them similarly to Definition 3, e.g., $a \approx b$ to $\{a\} \sqcap \text{Def}_a \sqsubseteq \{b\} \sqcap \text{Def}_b$ and vice versa and so on. This is sufficient for the construction of the interpretation J in the proof of Theorem 1 to go through. Note that the interpretations of constants (nominals) for which Def_a is not set in I_0 are irrelevant for constructing the interpretation J even though there could be assertions of the form $\top \sqsubseteq C$ that are applicable to such objects (one could even augment all such assertions by adding guards to avoid this effect). Therefore those constants (nominals) can be ignored completely by the reasoner and thus nodes corresponding to the constant symbols can be generated *lazily* on demand driven by the Def_a concept.

3.2 On Witnesses and Binary Absorption

Model building algorithms for checking the satisfaction of a concept C operate by manipulating an internal data structure (e.g., in the form of a node and edge labeled rooted tree with “back edges”). The data structure “encodes” a *partial description* of (eventual) interpretations \mathcal{I} for which $C^{\mathcal{I}}$ will be non-empty. Such a partial description will almost always abstract details on class membership for hypothetical elements of $\Delta^{\mathcal{I}}$ and on details relating to the interpretation of roles. To talk formally about absorption and lazy evaluation, it is necessary to codify

$$\begin{aligned}
& \{a\} \in \mathcal{L}^{\mathcal{W}}(x) \text{ and } \{a\} \in \mathcal{L}^{\mathcal{W}}(y) \text{ implies } x = y \\
& \{\{a\}, A\} \subseteq \mathcal{L}^{\mathcal{W}}(x), \text{ and } (\{a\} \sqcap A) \sqsubseteq C \in \mathcal{T}_u \text{ implies } C \in \mathcal{L}^{\mathcal{W}}(x) \\
& (x, y) \in R^{\mathcal{I}} \text{ and } \exists R.\top \sqsubseteq C \in \mathcal{T}_u \text{ implies } C \in \mathcal{L}^{\mathcal{W}}(x) \\
& (x, y) \in R^{\mathcal{I}} \text{ and } \exists R^{\neg}.\top \sqsubseteq C \in \mathcal{T}_u \text{ implies } C \in \mathcal{L}^{\mathcal{W}}(y) \\
& \{A_1, A_2\} \subseteq \mathcal{L}^{\mathcal{W}}(x) \text{ and } (A_1 \sqcap A_2) \sqsubseteq C \in \mathcal{T}_u \text{ implies } C \in \mathcal{L}^{\mathcal{W}}(x) \\
& A \in \mathcal{L}^{\mathcal{W}}(x) \text{ and } A \sqsubseteq C \in \mathcal{T}_u \text{ implies } C \in \mathcal{L}^{\mathcal{W}}(x) \\
& \neg A \in \mathcal{L}^{\mathcal{W}}(x) \text{ and } \neg A \sqsubseteq C \in \mathcal{T}_u \text{ implies } C \in \mathcal{L}^{\mathcal{W}}(x) \\
& C_1 \sqsubseteq C_2 \in \mathcal{T}_g \text{ implies } \neg C_1 \sqcup C_2 \in \mathcal{L}^{\mathcal{W}}(x) \\
& C_1 \doteq C_2 \in \mathcal{T}_g \text{ implies } \neg C_1 \sqcup C_2 \in \mathcal{L}^{\mathcal{W}}(x) \\
& C_1 \doteq C_2 \in \mathcal{T}_g \text{ implies } C_1 \sqcup \neg C_2 \in \mathcal{L}^{\mathcal{W}}(x)
\end{aligned}$$

Fig. 1: Absorption witness conditions

the idea of a partial description. This has been done in [5] by introducing the notion of a *witness*, of an interpretation that *stems* from a witness, and of what it means for a witness to be *admissible* with respect to a given terminology.

Definition 5. (Witness) Let C be an $\mathcal{ALC}\mathcal{IO}\mathcal{Q}(\mathbf{D})$ concept.¹ A witness $\mathcal{W} = (\Delta^{\mathcal{W}}, \cdot^{\mathcal{W}}, \mathcal{L}^{\mathcal{W}})$ for C consists of a non-empty set $\Delta^{\mathcal{W}}$, a function $\cdot^{\mathcal{W}}$ that maps NR to subsets of $\Delta^{\mathcal{W}} \times \Delta^{\mathcal{W}}$, and a function $\mathcal{L}^{\mathcal{W}}$ that maps $\Delta^{\mathcal{W}}$ to sets of $\mathcal{ALC}\mathcal{IO}\mathcal{Q}(\mathbf{D})$ concepts such that:

- (W1) there is some $x \in \Delta^{\mathcal{W}}$ with $C \in \mathcal{L}^{\mathcal{W}}(x)$,
- (W2) there is an interpretation \mathcal{I} that stems from \mathcal{W} , and
- (W3) for each \mathcal{I} that stems from \mathcal{W} , $x \in C^{\mathcal{I}}$ if $C \in \mathcal{L}^{\mathcal{W}}(x)$.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is said to stem from \mathcal{W} if $\Delta^{\mathcal{I}} = \Delta^{\mathcal{W}}$, $\cdot^{\mathcal{I}}|_{\text{NR}} = \cdot^{\mathcal{W}}$, for each $A \in \text{NC}$, $A \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \in A^{\mathcal{I}}$ and $\neg A \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \notin A^{\mathcal{I}}$, for each $a \in \text{NI}$, $\{a\} \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \in \{a\}^{\mathcal{I}}$ and $\neg\{a\} \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \notin \{a\}^{\mathcal{I}}$, for each $(f \text{ op } k)$, $(f \text{ op } k) \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \in (f \text{ op } k)^{\mathcal{I}}$ and $\neg(f \text{ op } k) \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \notin (f \text{ op } k)^{\mathcal{I}}$.

A witness \mathcal{W} is called *admissible with respect to a TBox \mathcal{T}* if there is an interpretation \mathcal{I} that stems from \mathcal{W} with $\mathcal{I} \models \mathcal{T}$.

The properties satisfied by a witness have been captured by the original lemmas 2.6 and 2.7 in [5]. We further extend binary absorption [6] to accommodate the absorbed ABox as shown in Section 3.

Definition 6. (Binary Absorption) Let $\mathcal{K} = \{\mathcal{T}, \mathcal{A}\}$ be a KB. A binary absorption of \mathcal{T} is a pair of TBoxes $(\mathcal{T}_u, \mathcal{T}_g)$ such that $\mathcal{T} \equiv \mathcal{T}_u \cup \mathcal{T}_g$ and \mathcal{T}_u contains axioms of the form $A_1 \sqsubseteq C$, $\neg A_1 \sqsubseteq C$, $\exists R.\top \sqsubseteq C$ (resp. $\exists R^{\neg}.\top \sqsubseteq C$), and the form $(A_1 \sqcap A_2) \sqsubseteq C$ and $(\{a\} \sqcap A) \sqsubseteq C$, where $\{A, A_1, A_2\} \subseteq \text{NC}$ and $a \in \text{NI}$.

A binary absorption $(\mathcal{T}_u, \mathcal{T}_g)$ of \mathcal{T} is called *correct* if it satisfies the following condition: For each witness \mathcal{W} and $x \in \Delta^{\mathcal{W}}$, if all conditions in Figure 1 are satisfied, then \mathcal{W} is admissible w.r.t. \mathcal{T} . A witness that satisfies the above property will be called *unfolded*.

¹ The definition of witness can be abstracted for any DLs that have $\mathcal{ALC}\mathcal{IO}$ as a sublanguage and that satisfy some criteria on the interpretations stated in [5].

The distinguishing feature of our extension of binary absorption is the addition of the first four implications in Figure 1. Binary absorption itself allows additional axioms in \mathcal{T}_u to be dealt with in a deterministic manner, as illustrated in our introductory example. ABox absorption, treating assertions as axioms, extends binary absorption to handle nominals in binary inclusion dependencies. In addition, domain and range constraints are also absorbed in a manner that resembles role absorption introduced in [13].

4 A Procedure for Indirect ABox Absorption

In this section, we present a procedure for ABox absorption, which extends binary absorptions [6]. Our algorithm also includes the possibility of absorbing domain and range constraints. The procedure prioritizes binary absorption to keep guarding constraints through restricted uses of nominals, yet it avoids guards for domain and range axioms by employing a variant of role absorption,

The algorithm is given a \mathcal{T}_K that consists of arbitrary axioms. It proceeds by constructing five TBoxes $\mathcal{T}_g, \mathcal{T}_{prim}, \mathcal{T}_{uinc}, \mathcal{T}_{binc}$, and \mathcal{T}_{rinc} such that: $\mathcal{T} \equiv \mathcal{T}_g \cup \mathcal{T}_{prim} \cup \mathcal{T}_{uinc} \cup \mathcal{T}_{binc} \cup \mathcal{T}_{rinc}$, \mathcal{T}_{prim} is primitive, \mathcal{T}_{uinc} consists of axioms of the form $A_1 \sqsubseteq C$, \mathcal{T}_{binc} consists of axioms of the form $(A_1 \sqcap A_2) \sqsubseteq C$ and $(\{a\} \sqcap A) \sqsubseteq C$ and none of the above primitive concept are defined in \mathcal{T}_{prim} , and \mathcal{T}_{rinc} consists of axioms of the form $\exists R.T \sqsubseteq C$ (or $\exists R^-.T \sqsubseteq C$). Here, \mathcal{T}_{uinc} contains unary inclusion dependencies, \mathcal{T}_{binc} contains binary inclusion dependencies and \mathcal{T}_{rinc} contains domain and range inclusion dependencies.

In the first phase, we move as many axioms as possible from \mathcal{T} into \mathcal{T}_{prim} . We initialize $\mathcal{T}_{prim} = \emptyset$ and process each axiom $X \in \mathcal{T}$ as follows.

1. If X is of the form $A \doteq C$, A is not defined in \mathcal{T}_{prim} , and $\mathcal{T}_{prim} \cup \{X\}$ is primitive, then move X to \mathcal{T}_{prim} .
2. If X is of the form $A \doteq C$, then remove X from \mathcal{T} and replace it with axioms $A \sqsubseteq C$ and $\neg A \sqsubseteq \neg C$.
3. Otherwise, leave X in \mathcal{T} .

In the second phase, we process axioms in \mathcal{T} , either by simplifying them or by placing absorbed components in \mathcal{T}_{uinc} , \mathcal{T}_{binc} or \mathcal{T}_{rinc} . We place components that cannot be absorbed in \mathcal{T}_g . We let $\mathbf{G} = \{C_1, \dots, C_n\}$ represent the axiom $\top \sqsubseteq (C_1 \sqcup \dots \sqcup C_n)$. Axioms are automatically converted to (out of) set notation. In addition, $\forall R.C$ (resp. $\forall R^-.C$) is considered a shorthand for $\exists^{\leq 0} R.C$ (resp. $\exists^{\leq 0} R^-.C$).

1. If \mathcal{T} is empty, then return the binary absorption $(\{A \sqsubseteq C, \neg A \sqsubseteq \neg C \mid A \doteq C \in \mathcal{T}_{prim}\} \cup \mathcal{T}_{uinc} \cup \mathcal{T}_{binc} \cup \mathcal{T}_{rinc}, \mathcal{T}_g)$.
Otherwise, remove an axiom \mathbf{G} from \mathcal{T} .
2. Simplify \mathbf{G} .
 - (a) If there is some $\neg C \in \mathbf{G}$ such that C is not a primitive concept, then add $(\mathbf{G} \cup \text{NNF}(\neg C) \setminus \{\neg C\})$ to \mathcal{T} , where the function $\text{NNF}(\cdot)$ converts concepts to negation normal form. Return to Step 1.

- (b) If there is some $C \in \mathbf{G}$ such that C is of the form $(C_1 \sqcap C_2)$, then add both $(\mathbf{G} \cup \{C_1\}) \setminus \{C\}$ and $(\mathbf{G} \cup \{C_2\}) \setminus \{C\}$ to \mathcal{T} . Return to Step 1.
 - (c) If there is some $C \in \mathbf{G}$ such that C is of the form $C_1 \sqcup C_2$, then apply associativity by adding $(\mathbf{G} \cup \{C_1, C_2\}) \setminus \{C_1 \sqcup C_2\}$ to \mathcal{T} . Return to Step 1.
3. Partially absorb \mathbf{G} .
- (a) If $\{\neg\{a\}, \neg A\} \subset \mathbf{G}$, and A is a guard, then do the following. If an axiom of the form $(\{a\} \sqcap A) \sqsubseteq A'$ is in \mathcal{T}_{binc} , add $\mathbf{G} \cup \{\neg A'\} \setminus \{\neg\{a\}, \neg A\}$ to \mathcal{T} . Otherwise, introduce a new concept $A' \in \text{NC}$, add $(\mathbf{G} \cup \{\neg A'\}) \setminus \{\neg\{a\}, \neg A\}$ to \mathcal{T} , and $(\{a\} \sqcap A) \sqsubseteq A'$ to \mathcal{T}_{binc} . Return to Step 1.
 - (b) If $\{\neg A_1, \neg A_2\} \subset \mathbf{G}$, and neither A_1 nor A_2 are defined in \mathcal{T}_{prim} , then do the following. If an axiom of the form $(A_1 \sqcap A_2) \sqsubseteq A'$ is in \mathcal{T}_{binc} , add $\mathbf{G} \cup \{\neg A'\} \setminus \{\neg A_1, \neg A_2\}$ to \mathcal{T} . Otherwise, introduce a new concept $A' \in \text{NC}$, add $(\mathbf{G} \cup \{\neg A'\}) \setminus \{\neg A_1, \neg A_2\}$ to \mathcal{T} , and $(A_1 \sqcap A_2) \sqsubseteq A'$ to \mathcal{T}_{binc} . Return to Step 1.
 - (c) If $\{\forall R.C\} = \mathbf{G}$ (resp. $\{\forall R^-.C\} = \mathbf{G}$), then do the following. Add $\exists R^-. \top \sqsubseteq C$ (resp. $\exists R. \top \sqsubseteq C$) to \mathcal{T}_{rinc} . Return to Step 1.
 - (d) If $\forall R. \neg A$ (resp. $\forall R^-. \neg A$) $\in \mathbf{G}$, then do the following. Introduce a new internal primitive concept A' and add both $A \sqsubseteq \forall R^-. A'$ (resp. $A \sqsubseteq \forall R. A'$) and $(\mathbf{G} \cup \{\neg A'\}) \setminus \{\forall R. \neg A\}$ (resp. $\setminus \{\forall R^-. \neg A\}$) to \mathcal{T} . Return to Step 1.
4. Unfold \mathbf{G} . If, for some $A \in \mathbf{G}$ (resp. $\neg A \in \mathbf{G}$), there is an axiom $A \doteq C$ in \mathcal{T}_{prim} , then substitute $A \in \mathbf{G}$ (resp. $\neg A \in \mathbf{G}$) with C (resp. $\neg C$), and add \mathbf{G} to \mathcal{T} . Return to Step 1.
5. Absorb \mathbf{G} . If $\neg A \in \mathbf{G}$ and A is not defined in \mathcal{T}_{prim} , add $A \sqsubseteq C$ to \mathcal{T}_{uinc} where C is the disjunction of $\mathbf{G} \setminus \{\neg A\}$. Return to Step 1.
6. If none of the above are possible (\mathbf{G} cannot be absorbed), add \mathbf{G} to \mathcal{T}_g . Return to Step 1.

Termination of our procedure can be established by a counting argument.

Theorem 2. *For any TBox \mathcal{T} , the ABox absorption algorithm computes a correct absorption of \mathcal{T} .*

Proof. The proof is by induction on iterations of our algorithm. We abbreviate the pair $(\{\mathcal{T}_{prim} \cup \mathcal{T}_{uinc} \cup \mathcal{T}_{binc} \cup \mathcal{T}_{rinc}, \mathcal{T}_g \cup \mathcal{T}\})$ as \mathcal{T} and claim that this pair is always a correct binary absorption. Initially, \mathcal{T}_{uinc} , \mathcal{T}_{binc} , \mathcal{T}_{rinc} and \mathcal{T}_g are empty, primitive axioms are in \mathcal{T}_{prim} , and the remaining axioms are in \mathcal{T} .

- In Step 3(a) or Step 3(b), \mathcal{T} is a correct absorption that derives from [6].
- In Step 3(c), \mathcal{T} is a correct absorption for domain and range constraints. The correctness proof of this step follows from Lemma 4.3 and 4.4 in [6].
- In Step 3(d), \mathcal{T} is a correct absorption by [6].
- In any of Steps 1, 2, 5-8, \mathcal{T} is a correct ABox absorption as they use only equivalence preserving operations.

Thus, \mathcal{T} is a correct binary absorption by induction.

5 Empirical Evaluation

The proposed absorption technique has been implemented in our CARE Assertion Retrieval Engine (CARE) with an underlying $\mathcal{ALCI}(\mathcal{D})$ DL reasoner. The DL reasoner has no additional ABox reasoning optimizations other than the technique presented here, and it has implemented only axiom absorption and blocking for TBox reasoning. All experiments were conducted on a MacBook with a 2.4GHz Intel Duo processor and 4GB RAM. All times, given in seconds, were averaged out over three independent runs for all reasoners and a reasoning timeout (denoted $-$) is set to 1500 seconds. Queries were posed over a suite of datasets (KBs) describing digital cameras. The KBs consist of digital camera specifications extracted from DPreview.com and pricing information from Amazon.com in which the seed KB, called DPC1, has one camera instance for each price found through Amazon for a camera model. The other KBs were generated from the seed KB by supplying n camera instances per price in DPC n . These KBs share the same TBox, i.e., 15 axioms, but have different ABox data, as shown in Figure 2a: it reports the number of individuals, concept, and role assertions, and the number of instances retrieved by each query over these datasets. Test queries are shown in Figure 3, which vary in query forms and selectivity.

	Inds	CAs	RAs	Q1	Q2	Q3	Q4		Q1	Q2	Q3	Q4
DPC1	5387	3225	7174	3	244	5	1426		-	-	-	-
DPC3	9711	3225	18672	9	732	11	1432	NG	-	-	-	-
DPC5	14035	3225	30170	15	1220	17	1438	PG	178.1	183.0	186.2	181.5
DPC7	18359	3225	41668	21	1708	23	1444	FG	6.5	6.0	6.1	6.1
DPC10	24845	3225	58915	30	2440	32	1453					

(a) Description of KBs

(b) Guarding Strategies (DPC1)

Fig. 2: Summary of results

Q1: $\exists hasInstance^-(Digital_SLR \sqcap (user_review = "5.00"))$
 Q2: $\exists hasInstance^-(Digital_SLR \sqcap \neg (user_review = "5.00"))$
 Q3: $\exists hasInstance^-(Digital_SLR \sqcap (user_review = "5.00")) \sqcup (price = "01400.00")$
 Q4: $\exists hasInstance^-(Digital_SLR \sqcap (user_review = "5.00")) \sqcup \neg (price = "01400.00")$

Fig. 3: Test queries

Query response times in Figure 2b compare different guarding strategies. Specifically, “No Guarding” (NG) is a straightforward implementation of the Tableaux algorithm (without assuming the consistency of the KB), “Partial Guarding” (PG) guards individuals so that only relevant individuals will be explored in Tableaux expansions, and “Full Guarding” (FG), in addition to “Partial Guarding,” further guards feature concepts that describe objects in the data so that only query-relevant feature concepts participate in reasoning. Figure 2b shows that CARE timed out under the NG strategy, while it managed to answer all queries under the PG strategy and efficiency increased substantially under the

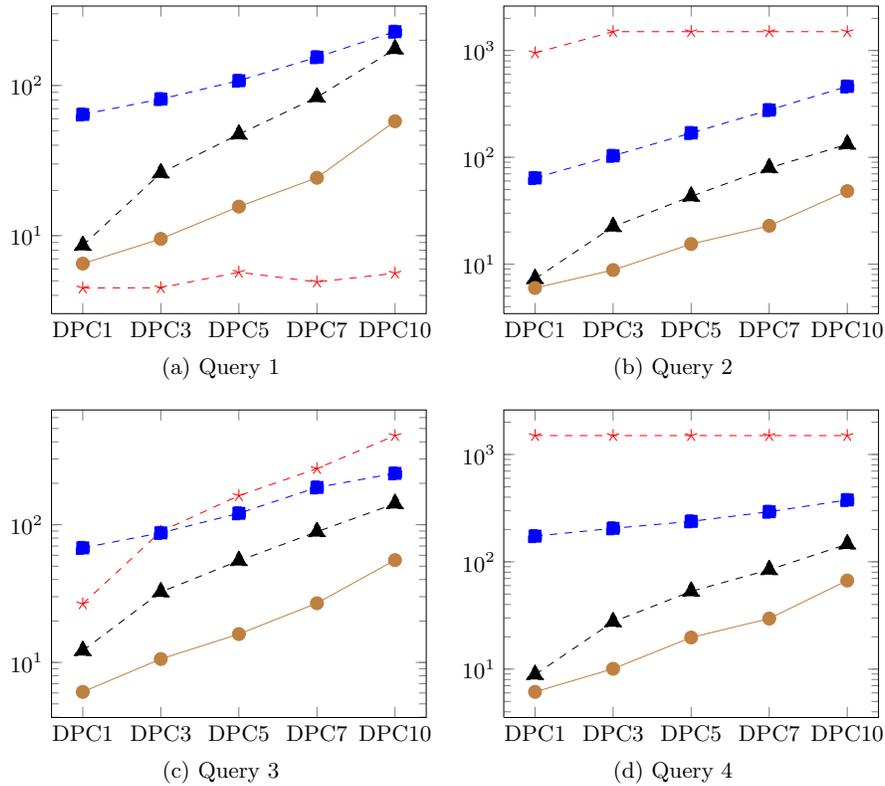


Fig. 4: Query response time comparison (logarithmic scale)

FG strategy. As numerous other optimization techniques have been developed (e.g., [2]) and implemented in most state-of-the-art reasoners, we juxtaposed their performance with CARE to show the efficacy of our proposed technique for ABox reasoning. All queries were posed via OWL API 3 for Pellet 2.3.0 and FaCT++ 1.5.3, and via JRacer in the form of nRQL for RacerPro 2.0. The query response time in Figure 4 does not consider the loading or preprocessing time for other reasoners, yet it includes the ABox absorption time (cf. Sect. 3) for CARE. The results show that CARE outperformed all other reasoners in all queries except Q1 (we believe this is due to deterministic precomputing at KB load time). Given that CARE is not as optimized as other reasoners, the results are significant.

6 Summary

We show how, with the presumption that a knowledge base is consistent, one can avoid considering irrelevant ABox individuals to the posed question while preserving soundness and completeness of answers to instance checking tasks.

Our experiments show that in realistic situations arising, e.g., in implementations of *assertion retrieval* [11] in which a number of instance checking queries are needed to answer a single user query, or in the case of *ontology-based query answering* [9, 7, 12, 8], when non-Horn DLs are used (and thus the above techniques cannot be applied), our technique makes querying often feasible. The experiments show, on relatively simple examples, that, while using the proposed technique allows answers to be computed in a few seconds, attempting the same tasks without the optimization is infeasible. To be effective, the technique relies on absorption procedures that have at least the capabilities of binary absorption. An interesting avenue of further work would be to explore how highly optimized DL reasoning procedures with more powerful capabilities for absorption such as procedures based on hypertableau [10] could further improve performance.

References

1. Baader, F., Franconi, E., Hollunder, B., Nebel, B., Profitlich, H.J.: An empirical analysis of optimization techniques for terminological representation systems, or: Making KRIS get a move on. *Applied Artificial Intelligence* 4, 109–132 (1994)
2. Haarslev, V., Möller, R.: On the scalability of description logic instance retrieval. *J. Autom. Reason.* 41(2), 99–142 (Aug 2008), <http://dx.doi.org/10.1007/s10817-008-9104-7>
3. Horrocks, I.: Using an expressive description logic: FaCT or Fiction? In: KR'98. pp. 636–647 (1998)
4. Horrocks, I., Tobies, S.: Optimisation of terminological reasoning. In: *Description Logics'00*. pp. 183–192 (2000)
5. Horrocks, I., Tobies, S.: Reasoning with axioms: Theory and practice. In: KR'00. pp. 285–296 (2000)
6. Hudek, A.K., Weddell, G.E.: Binary absorption in tableaux-based reasoning for description logics. In: *Description Logics'06* (2006)
7. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to query answering in DL-Lite. In: KR'10 (2010)
8. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to query answering in DL-Lite. In: *IJCAI'11*. pp. 2656–2661 (2011)
9. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic EL using a relational database system. In: *IJCAI'09*. pp. 2070–2075 (2009)
10. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. *J. Artif. Intell. Res. (JAIR)* 36, 165–228 (2009)
11. Pound, J., Toman, D., Weddell, G.E., Wu, J.: An assertion retrieval algebra for object queries over knowledge bases. In: *IJCAI'11*. pp. 1051–1056 (2011)
12. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR'10 (2010)
13. Tsarkov, D., Horrocks, I.: Efficient reasoning with range and domain constraints. In: *Description Logics'04* (2004)
14. Wu, J., Hudek, A., Toman, D., Weddell, G.: Assertion absorption in object queries over knowledge bases. In: KR'12 (2012)

A Parallel Reasoner for the Description Logic \mathcal{ALC}

Kejia Wu and Volker Haarslev

Concordia University, Montreal, Canada

Abstract. Multi-processor/core systems have become ubiquitous but the vast majority of OWL reasoners can process ontologies only sequentially. This observation motivates our work on the design and evaluation of *Deslog*, a parallel tableau-based description logic reasoner for \mathcal{ALC} . A first empirical evaluation for TBox classification demonstrates that *Deslog*'s architecture supports a speedup factor that is linear to the number of utilized processors/cores.

1 Introduction

The popularity of multi-processor/core computing facilities makes *Description Logic (DL)* reasoning feasible that scales w.r.t. the number of available cores.¹ Modern many-core architectures together with operating systems implementing symmetric multitasking efficiently support thread-level parallelism (TLP), where smaller subtasks are implemented as threads that are concurrently executed on available cores. In order to utilize such hardware for making DL reasoning scalable to the number of available cores, one has to develop new reasoning architectures efficiently supporting concurrency. Moreover, many well-known tableau optimization techniques need to be revised or adapted in order to be applicable in a parallel context. In this paper we present a new DL reasoning framework that fully utilizes TLP on many-core systems. In principle, standard tableau algorithms are well suited for parallelization because tableau completion rules usually do not depend on a sequential execution but require shared access to common data structures.

We consider the inherent non-determinism of DL tableaux as a feature of DL reasoning that naturally leads to parallel algorithms which are suitable for a shared-memory setting. For instance, a source for such a non-determinism are disjunctions and qualified cardinality restrictions for logics containing at least \mathcal{ALCQ} . Many standard DL reasoning services, e.g. concept satisfiability testing, TBox classification, instance checking, ABox realization, etc. might be amenable to be implemented in a non-deterministic way supporting parallelism. However, the use of parallelism in DL reasoning has yet to be explored systematically. Although it is advantageous to seek scalable solutions by means of parallelism, little progress has been made in parallel DL reasoning during the last decade [6]. Most DL reasoning algorithms and optimization techniques have only been investigated in a *sequential* context, but how these methods should be accommodated to parallelism remains mostly open. Besides these algorithmic and theoretical issues, the efficient implementation of a scalable parallel DL reasoner is also worth to be investigated. Many practical issues on shared-memory parallelism need to

¹ For ease of presentation we consider the terms processor and core as synonyms.

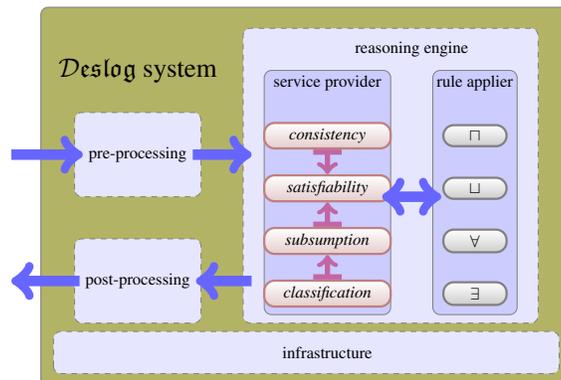


Fig. 1: The framework of *Deslog*

be researched, e.g., selecting suitable parallel architectures, designating efficient data structures and memory management, and exploring parallel algorithms.

In the remaining sections we introduce the design of *Deslog*, discuss related work, and present an evaluation demonstrating a mostly (super)linear scalability of *Deslog*.

2 Architecture of *Deslog*

2.1 Framework

The shared-memory parallel reasoner *Deslog* consists of three layers: (i) *pre-processing* layer, which converts the *Web Ontology Language (OWL)* representation of ontologies to internal data structures; (ii) *reasoning engine* layer, which performs the standard DL reasoning services and is composed of two key components, the *service provider* and the *tableau rule applier*; (iii) *post-processing* layer, which collects, caches, and saves reasoning results; (iv) *infrastructure* layer, which provides core components and utilities, such as structures representing concepts and roles, and the object copy tool. Figure 1 gives an overview of the framework.

First, OWL ontology data is read into the pre-processing layer. Various typical pre-processing operations, such as transformation into *negation normal form (NNF)*, axiom re-writing, and axiom absorption, are executed in this layer. The reasoner's run-time options, such as service selection, maximum number of threads, and rule application order, are also set up on this layer. We implemented this layer by using the OWL API [14]. The pre-processed data is streamed to the reasoning engine.

The reasoning engine performs primarily inference computation. The first key component of the reasoning engine is the service provider. As with popular DL reasoning systems, *Deslog* provides standard reasoning services, such as testing TBox consistency, concept satisfiability, etc. As we know, these services may depend on each other. In *Deslog*, the classification service depends on subsumption, and the latter depends on the satisfiability service. The service provider uses a suite of tableau-based algorithms

to perform reasoning. The reasoner adopts tableaux as its primary reasoning method, which is complemented by another key component, the tableau expansion rule applier. The main function of this component is to execute tableau rules in some order to build expansion forests. In *Deslog*, tableau expansion rules are designed as configurable plugins, so what rule has to be applied in what application order can be specified flexibly. At present, *Deslog* implements the standard *ALC* tableau expansion rules [4].

All three layers mentioned above use the facilities provided by the infrastructure layer. All common purpose utilities are part of this layer. For example, the threads manager, global counters, and the *globally unique identifier (GUID)* generator. In addition, the key data structures representing DL elements and basic operations on them are provided by this layer.

2.2 Key Data Structures

In contrast to popular DL reasoning systems, *Deslog* aims to improve reasoning performance by employing parallel computing, while data structures employed by sequential DL reasoners are not always suitable for parallelism.

Tree structures have been adopted by many tableau-based reasoners. However, a naive tree data structure introduces data races in a shared-memory parallel environment and are not well suited in a concurrency setting. Therefore, we need to devise more efficient data structures in order to reduce the amount of shared data as much as possible.

The new data structures facilitating concurrency must support DL tableaux. One important function of trees is to preserve non-deterministic branches generated during tableau expansion. Non-deterministic branches are mainly produced by the *disjunction rule* and the *at-most number restriction rules*. To separate non-deterministic branches into independent data vessels, which are suited to be processed in parallel, we adopt a list-based structure, called *stage*, to maintain a single non-deterministic branch, and a queue-based structure, called *stage pool*, to buffer all branches in a tableau. Every stage is composed of the essential elements of a DL ontology, concepts and roles.

As with any DL reasoner, the representation of concepts and roles are fundamental design considerations. The core data structure of *Deslog* is a four-slot list representing a *concept*. A *literal* uniquely identifies a distinct concept. An *operator* indicates the dominant DL constructor applied to a concept. Available constructors cover intersection, disjunction, existential and value restriction, and so on, and this slot can also be empty. The remaining two slots hold pointers to extend nested concept definitions, namely *left* and *right*. Figure 2 illustrates a DL concept encoded with the *Deslog* protocol.

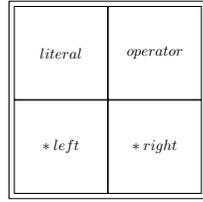


Fig. 2: *Deslog* data structure for a concept

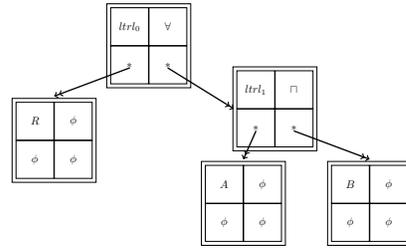
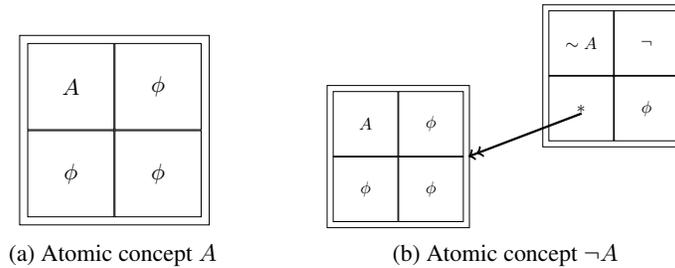


Fig. 3: *Deslog* data structure for the DL expression $\forall R.(A \sqcap B)$

Roles in *Deslog* are handled as a special type of concepts and have a similar structure as concepts. For instance, the encoding of the DL expression $\forall R.(A \sqcap B)$ is shown in Figure 3. Further properties needed for describing a role can be added to the generic structure, e.g., the number restriction quantity. A role data structure is also associated with a list recording instance pairs. With this design, DL concepts can be lined up seamlessly. Instances (i.e., labels in tableau expansions) are lists holding their typing data, concepts. There are also helper facilities, such as a role pool and an instance pool, which are useful to accelerate the indexing of objects.

A notable point on our encoding is how the *complement* of an atomic concept (i.e., concept name) is expressed (see Figures 4a and 4b).



(a) Atomic concept A

(b) Atomic concept $\neg A$

Fig. 4: *Deslog* data structure—atomic concepts

A principle of *Deslog*'s design is to model objects and behaviours involved in DL reasoning as independent abstractions as much as possible in order to facilitate concurrent processing. For instance, branches created during tableau expansion are encapsulated into standalone objects. Thus, a whole tableau expansion forest is designed as a list of branch objects. Tableau expansion rules and even some key optimization techniques are also designed as independent components.

2.3 Implementation

As aforementioned, multi-processor computers are becoming the main stream, it is expected that idle processors are utilized, and shared-memory TLP is quite suitable for this purpose.

One significant aspect of this research is to investigate how well important DL reasoning optimization techniques are suited to be implemented in a parallel reasoner and how they should be adapted if plausible. *Deslog* has adopted the following optimization techniques.

1. **Lazy-unfolding** This technique enables a reasoner to unfold a concept only when necessary [5].
2. **Axiom absorption** *Disjunctive branches* introduced by naively internalizing TBox-axioms is one of the primary sources of reasoning inefficiency. With the axiom absorption technique, a TBox is separated into two parts, the *general* TBox and the *unfoldable* TBox. Then, using internalization to process the general part and lazy-unfolding to process the unfoldable part can reduce reasoning time dramatically [15, 25].
3. **Semantic branching** This DPLL style technique prunes disjunctive branches by avoiding to compute the same problem repeatedly [15].

Other primary optimization techniques, such as *dependency directed backtracking* [4, Chapter 9] [15] and *model merging* [12] are currently being implemented. It is noticeable that not all significant optimization techniques are suitable for concurrency. Some of them still depend on complex shared data and may significantly degrade the performance of a concurrent program. Based on these elemental techniques, we completed a suite of standard TBox reasoning services.

The current system implements a parallel *ALC* TBox classifier. It can concurrently classify an *ALC* terminology. The parallelized classification service of *Deslog* computes subsumptions in a brutal way [5]. It is obvious that the algorithm is sound and complete and has order of n^2 time complexity in a sequential context. In order to figure out a terminology hierarchy, the algorithm calculates the subsumptions of all atomic concepts pairs. A subsumption relationship only depends on the involved concepts pair, and does not have any connections with the computation order. Therefore, the subsumptions can be computed in parallel, and soundness and completeness are retained in a concurrent context.

A rather difficult issue in implementing a parallel DL reasoner is managing overhead. This issue is relatively easy for high level parallel reasoning, where multiple threads mainly execute reading operations on some shared data, thus, we implemented the parallel classification service first.

Besides the high-level parallelized service, classification, low level parallelized processing is being developed. In the architecture of *Deslog*, the classification service uses subsumption, and subsumption uses satisfiability. The low level parallel reasoning focuses on dealing with non-deterministic branches, which are represented as stages in *Deslog*.

It might seem easy to process stages in parallel, but quite some effort is required to achieve a satisfying scalability via concurrency. The first noticeable fact is that from

a root stage every stage may generate new ones. At present, our strategy is using one thread to process one stage. That means the stages buffer, the stage pool, is frequently accessed by multiple threads. That accessing includes both writing and reading shared data frequently. So, designing a high-performance stage buffer and efficient accessing schemes is an essential condition for the enabling a of scalable performance improvement. Otherwise, a parallel approach only causes overhead instead of performance improvement. We are currently working on efficient low-level parallel reasoning.

Although there are robust shared-memory concurrent libraries available, such as the C++ *Boost.Thread* library and the Java *concurrent* package, according to our experience, using these libraries immoderately often degrades performance. Therefore, one needs to design sophisticated structures which better avoid shared data, or which do not access shared data frequently.

3 Evaluation

Deslog is implemented in Java 6 in conformity with the aforementioned design. The *parallelism* of *Deslog* is based on a *multi-threading model* and aims at exploiting *symmetric multiprocessing (SMP)* supported by *multi-processor computing* facilities. The system is implemented in Java 6 for Java's relatively mature parallel ecosystem.² Specifically, the *java.util.concurrent* package of Java 6 is utilized.

In the following we report on experiments to demonstrate that a shared-memory parallel tableau-based reasoner can achieve a scalable performance improvement.

3.1 Conducted Experiments

The classification service of *Deslog* can be executed concurrently by multiple threads. We conducted a group of tests, and they show that *Deslog* has an obvious scalability.

All tests were conducted on a 16-core computer running Solaris OS and Sun Java 6. Many of the test cases were chosen from *OWL Reasoner Evaluation Workshop 2012 (ORE 2012)* data sets. We manually changed the expressivity of some test cases to *ALC* so that *Deslog* could reason about them. Table 1 lists the metrics of the test cases. The results are shown in Figures 5-7.

3.2 Discussion

The experimental results collected from testing ontologies show a scalable performance improvement. The tests on more difficult ontologies demonstrate a better scalability due to reduced overhead.

Because the computing time of the single thread configuration, T_1 , is rather small for some ontologies, often smaller than ten seconds, the overhead introduced by maintaining multiple threads can limit the scalability. For some of these ontology tests, the reasoning times are reduced to several milliseconds, i.e., the whole work load assigned to a single thread is around several milliseconds in these settings. According to our empirical results, a small work load w.r.t. the overhead, which is produced by manipulating

² All components and sources of the system are available at <http://code.google.com/p/deslog/>.

Table 1: Characteristics of the used test ontologies

Ontology	DL expressivity	Concept count	Axiom count
bfo	\mathcal{ALC}	36	45
pharmacogenomics_complex	\mathcal{ALC}	145	259
economy	$\mathcal{ALCH}(\mathcal{D})$	339	563
transportation	$\mathcal{ALCH}(\mathcal{D})$	445	489
mao	$\mathcal{ALE}+$	167	167
yeast_phenotype	\mathcal{AL}	281	276
loggerhead_nesting	\mathcal{ALE}	311	347
spider_anatomy	\mathcal{ALE}	454	607
pathway	\mathcal{ALE}	646	767
amphibian_anatomy	$\mathcal{ALE}+$	703	696
flybase_vocab	$\mathcal{ALE}+$	718	726
tick_anatomy	$\mathcal{ALE}+$	631	947
plant_trait	\mathcal{ALE}	976	1140
evoc	\mathcal{AL}	1001	990
protein	$\mathcal{ALE}+$	1055	1053

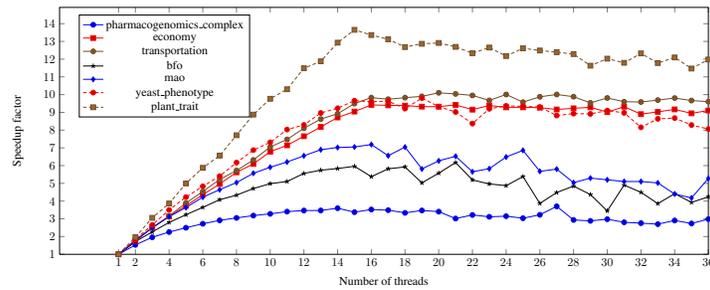


Fig. 5: Speedup factor for pharmacogenomics_complex, economy, transportation, bfo, mao, yeast_phenotype, and plant_trait

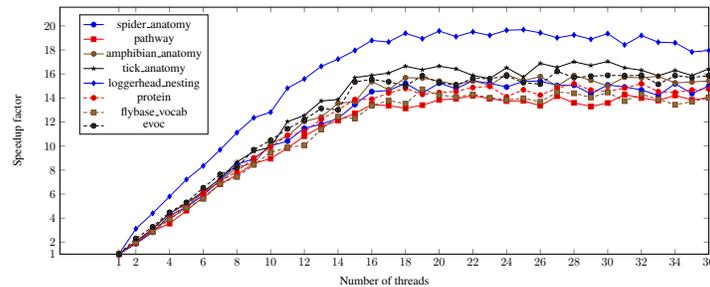


Fig. 6: Speedup factor for spider_anatomy, pathway, amphibian_anatomy, tick_anatomy, loggerhead_nesting, protein, flybase_vocab, and evoc

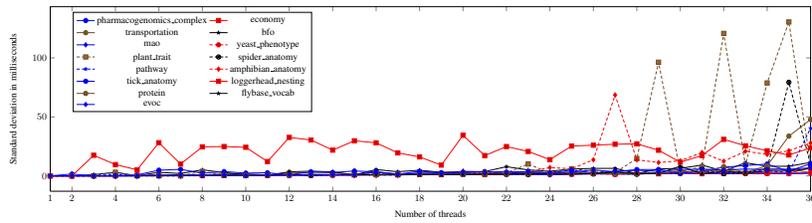


Fig. 7: The standard deviations of thread runtimes

threads as well as accessing shared data, counteracts the benefits gained from parallel processing and the reasoning performance starts to degrade.

When testing ontologies that are big enough, the observed scalability is linear, sometimes even superlinear. These bigger ontologies need longer single thread computing times (T_1). The overhead introduced by maintaining a *tolerable* number of multiple threads is very small and becomes insignificant. A tolerable number, N_i , should always be smaller than or equal to the total number of available processors. In our conducted tests due to the available hardware we have $N_i \in [1, 16]$ but in order to stress multi-threading we conducted all tests with up to 32 threads. In some cases, even though the number of threads exceeds 16, the reasoning performance keeps stable in a rather long run. This clearly supports our hypothesis that a further scalability improvement could be achieved by adding more processors, and we will verify this hypothesis when better experimental hardware becomes available.

Figure 7 shows the standard deviation of thread runtimes measured in the series of tests (in the unit of milliseconds). Overall, the deviations are limited to an acceptable range, i.e., below 140 milliseconds, which is relatively insignificant w.r.t. system overhead. This implies that the work load is well balanced among threads. That is to say, all threads are as much busy as possible. For the most part, when the number of threads is smaller than the tolerable number, 16, deviations are normally close to 0. When threads are added beyond 16, deviations become greater. This is because some processors execute more than one thread, and hereby the thread contexts switching produces a lot of overhead. In our original implementation, we had distributed all subsumption candidates into independent lists, every of which mapped to a thread, but the deviations were sometimes too large. So, the *Deslog* classification uses now a shared queue to buffer all subsumption candidates, in order keep all threads busy.

We had conducted similar experiments on a high-performance computing cluster, and the results were rather disappointing. The speedup factor gained on the cluster was generally below 3 although we assigned at least 16 processors for each test. The most plausible explanation we can give is that the complex hardware and software environment of the cluster degrades the performance of *Deslog*. The cluster consists of three types of computing nodes with respect to the built-in processors: 4-core, 8-core, and 16-core. And the same type of computers may have heterogeneous architectures. A job is scheduled on one or more computers randomly. It is normal that a job is assigned to more than one computer, and the communication between computers results in a bottleneck. Another possible reason is that the cluster does not guarantee exclusive usage,

which means it is possible that more than one job is running on the same computer at the same time.

Deslog can only deal with \mathcal{ALC} ontologies at present, and we expect that more interesting results will be obtained by implementing more powerful tableau expansion rules (e.g., see [16]). We already investigated the feasibility of several tableau optimization techniques for a concurrency setting, but most of them are not yet fully implemented or tested.

4 Related Work

Our research investigates the potential contribution of concurrent computing to tableau-based DL reasoning. Tableau-based DL reasoning has been extensively researched in sequential contexts. A large amount of literature is available for addressing sequential DL reasoning (see [4]). Only a few approaches investigated parallel DL reasoning so far.

The work in [19] reports on a parallel \mathcal{SHN} reasoner. This reasoner implemented parallel processing of *disjunctions* and *at-most cardinality restrictions*, as well as some DL tableau optimization techniques. The experimental results show noticeable performance improvement in comparison with sequential reasoners. This work mainly showed the feasibility of parallelism for low level tableau-based reasoning, and did not mention high level reasoning tasks, such as classification. Besides utilizing non-determinism, this research also presented the potential of making use of *and-parallelism*, and we plan to follow up on this idea.

The canonical top-search algorithm, as well as its dual bottom-search, can be executed in parallel, but extra work is needed to preserve completeness. Such an approach is presented in [1, 2] and the experimental results are very promising and demonstrate the feasibility of parallelized DL reasoning.

In [17, 23] a *consequence-based* DL reasoning method was proposed, mainly dealing with Horn ontologies. Based on consequence-based reasoning and the results of [3], the work in [18] reports on a highly optimized reasoner that can classify \mathcal{EL} ontologies *concurrently*.

Two hypotheses on parallelized ontology reasoning were proposed in [6]: *independent ontology modules* and a *parallel reasoning algorithm*. *Independent ontology modules* strive for structuring ontologies as modules which naturally can be computed in parallel. The idea of partitioning ontologies into modules is supported by [11, 9, 10]. According to the second hypothesis, extensive research on parallelized logic programming does not contribute much to DL reasoning. Furthermore, some DL fragments, without disjunction and at-most cardinality restriction constructors, do not profit much from parallelizing non-deterministic branches in tableau expansion.

The research mentioned above is focusing on a shared-memory multi-threading environment. There is also quite some research proposing distributed solutions, and some of these ideas are also worth being tried in a shared-memory environment.

In [21] the idea of applying a *constraint programming* solver, *Mozart*, was proposed to \mathcal{ALC} tableau reasoning in parallel, and the implementation was reported on in [20]. The experimental results show scalability to some extent.

Recently, some research work focuses on how DL reasoning can be applied to *Resource Description Framework (RDF)* and *OWL*. This trend leads to research on reasoning about massive web ontologies in a scalable way. MapReduce [22], ontology mapping [13], ontology partitioning [10], rule partitioning [24], *distributed hash table (DHT)* [8], swarm intelligence [7], etc., are examples of these approaches. However, we do not consider these approaches as relevant in our context.

5 Conclusion and Future Work

DL has been successfully applied in many domains, amongst which utilizing knowledge on the Internet has become a research focus in recent years. *Scalable* solutions are required for processing an enormous amount of structured information spreading over the Internet. Meanwhile, multi-processor computing facilities are becoming the main stream. Thus, we consider research on parallelism in DL reasoning as necessary to utilize available processing power.

The objective of this research is to explore how parallelism plays a role in tableau-based DL reasoning. A number of tableau-based DL reasoning optimization techniques have been extensively researched, but most of them are investigated in *sequential* contexts, so adapting these methods to the *parallel* context is an important part of this research.

We have partially shown that shared-memory parallel tableau-based DL reasoning can contribute to scalable solutions. This paper introduced our reasoner, *Deslog*, of which the architecture is devised specially for a shared-memory parallel environment. We presented an aspect of the reasoner's concurrency performance, and a good scalability could be demonstrated for TBox classification.

In the near future, we will enhance *Deslog* so that it can reason about more expressive DL languages. More powerful tableau expansion rules will be added. Concurrency-suitable tableau optimization techniques will be implemented. Moreover, we plan to investigate non-determinism more closely, and to design corresponding parallel algorithms.

References

1. Aslani, M., Haarslev, V.: Towards parallel classification of TBoxes. In: Proceedings of the 21st International Workshop on Description Logics (DL2008), Dresden, Germany. vol. 353 (2008)
2. Aslani, M., Haarslev, V.: Parallel TBox classification in description logics—first experimental results. In: Proceeding of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence. pp. 485–490 (2010)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: International Joint Conference on Artificial Intelligence. vol. 19, p. 364 (2005)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The description logic handbook: theory, implementation, and applications. Cambridge University Press (2003)

5. Baader, F., Hollunder, B., Nebel, B., Profitlich, H.J., Franconi, E.: An empirical analysis of optimization techniques for terminological representation systems. *Applied Intelligence* 4(2), 109–132 (1994)
6. Bock, J.: Parallel computation techniques for ontology reasoning. In: *Proceedings of the 7th International Conference on The Semantic Web*. pp. 901–906 (2008)
7. Dentler, K., Guéret, C., Schlobach, S.: Semantic web reasoning by swarm intelligence. In: *The 5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*. p. 1 (2009)
8. Fang, Q., Zhao, Y., Yang, G., Zheng, W.: Scalable distributed ontology reasoning using DHT-based partitioning. In: *The semantic web: 3rd Asian Semantic Web Conference, ASWC 2008*. pp. 91–105 (2008)
9. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: A logical framework for modularity of ontologies. In: *Proc. IJCAI*. pp. 298–304 (2007)
10. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research* 31(1), 273–318 (2008)
11. Grau, B.C., Parsia, B., Sirin, E., Kalyanpur, A.: Modularizing OWL ontologies. In: *K-CAP 2005 Workshop on Ontology Management* (2005)
12. Haarslev, V., Möller, R., Turhan, A.Y.: Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. *Automated Reasoning 2083/2001*, 61–75 (2001)
13. Homola, M., Serafini, L.: Towards formal comparison of ontology linking, mapping and importing. In: *23rd International Workshop on Description Logics, DL2010*. p. 291 (2010)
14. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. *Semantic Web* 2(1), 11–21 (2011)
15. Horrocks, I.: Optimising tableaux decision procedures for description logics. Ph.D. thesis, The University of Manchester (1997)
16. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic SHIQ. In: *Automated deduction-CADE-17: 17th International Conference on Automated Deduction, Pittsburgh, PA, USA, June 17-20, 2000: proceedings*. vol. 17, p. 482 (2000)
17. Kazakov, Y.: Consequence-driven reasoning for horn SHIQ ontologies. In: *Proceedings of the 21st international joint conference on Artificial intelligence*. pp. 2040–2045 (2009)
18. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of EL ontologies. In: *Proceedings of the 10th International Semantic Web Conference* (2011)
19. Liebig, T., Müller, F.: Parallelizing tableaux-based description logic reasoning. In: *Proceedings of the 2007 OTM Confederated International Conference on the Move to Meaningful Internet Systems-Volume Part II*. pp. 1135–1144 (2007)
20. Meissner, A.: A simple parallel reasoning system for the ALC description logic. In: *Computational Collective Intelligence: Semantic Web, Social Networks and Multiagent Systems (First International Conference, ICCCI 2009, Wroclaw, Poland, 2009)*. pp. 413–424. Springer (2009)
21. Meissner, A., Brzykcy, G.: A parallel deduction for description logics with ALC language. *Knowledge-Driven Computing* 102, 149–164 (2008)
22. Mutharaju, R., Maier, F., Hitzler, P.: A MapReduce algorithm for EL+. In: *23rd International Workshop on Description Logics DL2010*. p. 456 (2010)
23. Simančík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011)* (2011)
24. Soma, R., Prasanna, V.K.: Parallel inferencing for OWL knowledge bases. In: *Proceedings of the 2008 37th International Conference on Parallel Processing*. pp. 75–82 (2008)
25. Wu, J., Haarslev, V.: Planning of axiom absorptions. In: *Proceedings of International Workshop on Description Logics 2008* (2008)

Towards an Expressive Decidable Logical Action Theory

Wael Yehia¹ and Mikhail Soutchanski²

¹ Department of Computer Science and Engineering York University, 4700 Keele Street,
Toronto, ON, M3J 1P3, Canada
w2yehia@cse.yorku.ca

² Department of Computer Science, Ryerson University, 245 Church Street, ENG281, Toronto,
ON, M5B 2K3, Canada mes@scs.ryerson.ca

Introduction

The projection problem is an important reasoning task in AI. It is a prerequisite to solving other computational problems including planning and high-level program execution. Informally, the projection problem consists in finding whether a given logical formula is true in a state that results from a sequence of transitions, when knowledge about an initial state is incomplete. In description logics (DLs) and earlier terminological systems, this problem was formulated using roles to represent transitions and concept expressions to represent states. This line of research as well as earlier applications of DLs to planning and plan recognition are discussed and reviewed in [5, 11] to mention a few only. Using a somewhat related approach, the projection problem and a solution to the related frame problem (i.e., how to provide a concise axiomatization of non-effects of actions) have been explored using propositional dynamic logic, e.g., see [10, 9]. These papers discuss relations with the propositional fragment of the situation calculus and review previous work. A more recent work explores decidable combinations of several modal logics, or combining description logics with a modal logic of time or with a propositional dynamic logic [1, 23, 7]. The resulting logics are somewhat limited in terms of expressivity because to guarantee the decidability of the satisfiability problem in the combined logic, only atomic actions can be allowed. In applications, it is sometimes convenient to consider actions with arbitrary many arguments.

On the other hand, there are several proposals regarding the integration of DLs and reasoning about actions [19, 4, 17, 6, 13]. In [13], it is shown that the projection problem is decidable in a proposed fragment of the situation calculus (SC). However, the logical languages developed in these papers are not expressive enough to represent some of the action theories popular in AI or to solve the projection problem in a general case. For example, Gu& Soutchanski propose a DL based situation calculus [13], where the projection problem is reduced to the satisfiability problem in $\mathcal{ALCO}(U)$, a DL that adds nominals O and the universal (global) role U to the well known description logic \mathcal{ALC} . (The universal role is interpreted as the binary relation that links any two domain elements.) They consider Reiter's basic action theories (BATs) [22], but impose syntactic constraints on the formulas that can appear in axioms by concentrating on a subset FO_{DL} of FO^2 formulas, where FO^2 is a fragment of first order logic (FOL) with only two variables. In the fragment of SC that they consider, action functions may have at most two object arguments, the formulas in the precondition axioms (PA) and context formulas in the successor state axioms (SSA) should be FO_{DL} formulas (if the situation argument is suppressed), where FO_{DL} formulas are those FO^2 formulas, which can be translated into a concept in $\mathcal{ALCO}(U)$ using the standard translation between DLs and fragments of FOL. They illustrate their proposal with several realistic examples of

dynamic domains, but it turns out that some of the well-known examples, e.g., the Logistics domain from the first International Planning Competition (IPC) [20], cannot be represented due to syntactic restrictions on the language they consider. Here and subsequently, when we mention planning domain specifications, we consider them as FOL theories without making the Domain Closure Assumption (DCA) common in planning, i.e., without reducing them to purely propositional level. Later, [12] introduces a possible extension, where the syntactic restrictions on the class of formulas FO_{DL} are relaxed, but stipulates SSAs for dynamic roles (fluents with two object arguments and one situational argument) to be context-free. She conjectures, but does not prove, that the projection problem in her extension can be reduced to satisfiability in $\mathcal{ALCO}(\mathcal{U})$.

In our paper, we consider an even more expressive fragment of SC, called \mathcal{P} , where all SSAs can be context dependent with context conditions formulated in a language \mathcal{L} that includes FO_{DL} as a proper fragment. Manual translations of planning specifications (from IPC) into our fragment \mathcal{P} show that \mathcal{P} has expressive power sufficient to represent not only Blocks World and Logistics, but also many other domains. In any case, reducing projection to satisfiability in $\mathcal{ALCO}(\mathcal{U})$ is justified by the fact that there are several off-the-shelf OWL2 reasoners that can be employed to solve the latter problem, since a DL $SR\mathcal{OIQ}$ underlying the Web Ontology Language (OWL2) includes $\mathcal{ALCO}(\mathcal{U})$ as a fragment [8]. In our paper, we concentrate on foundational work and explore the logical properties of \mathcal{P} . Our paper contributes by formulating an expressive fragment of SC where the projection problem is decidable without DCA and closed world assumption (CWA), i.e., when an initial theory is incomplete and is not purely propositional. We hope that research outlined in our paper will attract the description logic community to interesting research issues on the boundary between DLs and reasoning about actions.

Definition of \mathcal{P}

We assume that the reader is familiar with SC from [21, 22] and knows that a BAT $\mathcal{D} = \mathcal{D}_{AP} \cup \mathcal{D}_{SS} \cup \mathcal{UNA} \cup \mathcal{D}_{S_0} \cup \Sigma$ consists of the precondition axioms (PAs) \mathcal{D}_{AP} that use the binary predicate symbol $Poss$, successor state axioms (SSAs) \mathcal{D}_{SS} , a set of unique name axioms \mathcal{UNA} , an initial theory \mathcal{D}_{S_0} that specifies an incomplete theory of the initial situation S_0 , and Σ - a set of domain independent foundational axioms about the relation $s_1 \preceq s_2$ of precedence between situations s_1 and s_2 . In [22], axioms Σ are formulated in second-order logic, all other axioms are formulated in FOL, so we assume the usual definitions of sorts, terms, well-formed formulas, and so on. A fluent is a predicate with the last argument s of sort situation. As usual, we say that a situation calculus FOL formula $\psi(s)$ is *uniform* in s , if s is the only situation term mentioned in $\psi(s)$, the formula ψ has no occurrences of the predicates $Poss$, \prec , and has no quantifiers over variables of sort situation. The formula ψ obtained by deleting all arguments s from fluents in the formula $\psi(s)$ uniform in s is called the formula with *suppressed* situation argument; the interested reader can find details in [21].

Fluents with a single object argument, $F(x, s)$, are called *dynamic concepts*, and fluents with two object arguments, $F(x, y, s)$, are called *dynamic roles*. In the signature of a BAT \mathcal{D} , any predicate that is not a fluent must have either one or two arguments, and is called either a *(static) concept*, or a *(static) role*, respectively. Subsequently, we

consider only BATs with relational fluents, and do not allow any other function symbols except $do(a, s)$ and action functions $A(x)$. In particular, terms of sort object can be only constants or variables. Each action function can have any number of object arguments.

To specify syntactic constraints on \mathcal{D}_{ap} and \mathcal{D}_{ssa} , we consider a language \mathcal{L} , that has at most $n + 2$ object variables x, y, z_1, \dots, z_n , for some integer $n > 0$. We assume \mathcal{L} has at least n constants $b_i, 1 \leq i \leq n$. The purpose of the variables z_i is to serve as place-holders to be instantiated with constants b_i that occur as named object arguments of ground action terms. This language \mathcal{L} consists of two related sets of formulas: \mathbf{F}^x and \mathbf{F}^y . Formulas $\phi(x)$ from the set \mathbf{F}^x can have as free variables either x , or some of the place-holder variables $z_i, 1 \leq i \leq n$, but cannot have free occurrences of y . Formulas $\phi(y)$ from the set \mathbf{F}^y can have free occurrences of either y , or some of the place holders $z_i, 1 \leq i \leq n$, but cannot have free occurrences of x . Note the formulas ϕ may have free variables z_i that are not shown explicitly, but it will be always clear from the context which variables are free in the formulas. We use the symbol $\tilde{\cdot}$ to denote a bijection between \mathbf{F}^x and \mathbf{F}^y . If $\phi(x) \in \mathbf{F}^x$, then $\tilde{\phi}(y)$ is the *dual* formula of $\phi(x)$, obtained by renaming in $\phi(x)$ every occurrence of x (both free and bound) with y and every bound occurrence of y with x . Similarly, if $\phi(y) \in \mathbf{F}^y$, then $\tilde{\phi}(x)$ is the *dual* formula to $\phi(y)$ obtained by replacing every occurrence of y with variable x , and every bound occurrence of x with y . The sets \mathbf{F}^x and \mathbf{F}^y have a non-empty intersection. For example, sentences that mention constants only, and \mathbf{F}^x formulas that have only occurrences of z variables belong to both \mathbf{F}^x and to \mathbf{F}^y . Each formula ϕ without x, y variables is mapped by bijection $\tilde{\phi}$ to itself. We are ready to give the following inductive definition.

Definition 1. Let \mathcal{L} be the set of first-order logic formulas such that $\mathcal{L} = \mathbf{F}^x \cup \mathbf{F}^y$, and $\tilde{\cdot}$ be a bijection between formulas in \mathbf{F}^x and \mathbf{F}^y as defined above, where the sets \mathbf{F}^y and \mathbf{F}^x are minimal sets constructed as follows. (We focus on \mathbf{F}^x , since \mathbf{F}^y is similar.)

1. \top and \perp are in \mathbf{F}^x .
2. If AC is a unary predicate symbol, z is a variable distinct from x and y , and b is a constant, then the formulas $AC(x)$, $AC(z)$, and $AC(b)$ are in \mathbf{F}^x .
3. If b is a constant, and z is a variable that is distinct from x and y , then the formulas $x = x$, $x = b$, $x = z$ are in \mathbf{F}^x .
4. If R is a binary predicate symbol, b_1 and b_2 are constants, and z_1 and z_2 are variables that are distinct from x and y , then $R(z_1, z_2)$, $R(b_1, b_2)$, $R(b_1, z_2)$, $R(z_1, b_2)$, $R(x, b_2)$ and $R(x, z_2)$ are formulas in \mathbf{F}^x .
5. If $\phi \in \mathbf{F}^x$, then also $\neg\phi \in \mathbf{F}^x$.
6. If $\phi, \psi \in \mathbf{F}^x$, then both $(\phi \wedge \psi) \in \mathbf{F}^x$ and $(\phi \vee \psi) \in \mathbf{F}^x$.
7. If $\phi(x) \in \mathbf{F}^x$, R is a binary predicate symbol, b is any constant, z is any variable distinct from x and y , and $\tilde{\phi}(y)$ is the formula dual to $\phi(x)$, then all of the following formulas with quantifiers guarded by R belong to \mathbf{F}^x : $\exists y.R(x, y) \wedge \tilde{\phi}(y)$, $\exists y.R(b, y) \wedge \tilde{\phi}(y)$, $\exists y.R(z, y) \wedge \tilde{\phi}(y)$, as well as $\forall y.R(x, y) \supset \tilde{\phi}(y)$, $\forall y.R(b, y) \supset \tilde{\phi}(y)$, $\forall y.R(z, y) \supset \tilde{\phi}(y)$.
8. If $\phi \in \mathbf{F}^x$, $\tilde{\phi}$ is the formula dual to ϕ , then $[\exists x].\phi(x)$, $[\forall x].\phi(x)$ as well as $[\exists y].\tilde{\phi}(y)$, $[\forall y].\tilde{\phi}(y)$ belong to \mathbf{F}^x , where $[\exists]$ ($[\forall]$, respectively) means that quantifiers are optional and applied only when a formula has a free variable.

The intuition behind the definition of \mathcal{L} is that any variable z other than x and y has to be free in a formula from \mathcal{L} . The set of formulas $FO_{DL} = FO_{DL}^x \cup FO_{DL}^y$ defined in [13]

is a proper subset of \mathcal{L} because the set of formulas FO_{DL}^x (FO_{DL}^y , respectively) is a proper subset of \mathbf{F}^x (\mathbf{F}^y , respectively): no place holder variables z_1, \dots, z_n are allowed in FO_{DL}^x and FO_{DL}^y . We say a formula $\phi \in \mathcal{L}$ is a z -free \mathcal{L} formula, if all occurrences of variables z (if any), other than x and y , in ϕ are instantiated with constants.

Lemma 1. *There are syntactic translations between the set of z -free formulas $\phi \in \mathcal{L}$ and the concept expressions from the language $\mathcal{ALCC}(U)$ in both directions, i.e., they are equally expressive. Moreover, such translations lead to no more than a linear increase in the size of the translated formula.*

This lemma is proved using the standard translation between DLs and FOL; the proof is similar to the proof of Lemma 1 in [13]. Using the fluents $Loaded(box, s)$, $At(box, city, s)$, and $In(box, vehicle, s)$ from Logistics as an example, after suppressing s , a z -free \mathcal{L} formula $Loaded(B_1) \vee \exists x (Box(x) \wedge x \neq B_1 \wedge In(x, T_1))$ is translated as $\exists U. (\{B_1\} \sqcap Loaded) \sqcup \exists U. (Box \sqcap \neg\{B_1\} \sqcap \exists In. \{T_1\})$, where $\{B_1\}$, $\{T_1\}$ are nominals (i.e., concepts interpreted as singleton sets), and $\forall x (\neg Box(x) \vee x = B_1 \vee At(x, Toronto))$, all boxes distinct from B_1 are in Toronto, is translated as $\forall U. (\neg Box \sqcup \{B_1\} \sqcup \exists At. \{Toronto\})$. Notice why nominals and U are important. Subsequently, we consider BATs that use in axioms \mathcal{L} -like formulas uniform in s . This motivates the following requirements. For brevity, let a vector \mathbf{x} of object variables denote either x , or y , or $\langle x, y \rangle$; also, let \mathbf{z} denote a finite vector of place holder variables. *Action precondition axioms* \mathcal{D}_{AP} : For each action function $A(\mathbf{z})$, there is a single precondition axiom uniform in s :

$$(\forall \mathbf{z}, s). Poss(A(\mathbf{z}), s) \equiv \Pi_A(\mathbf{z}, s), \quad (1)$$

where $\Pi_A(\mathbf{z}, s)$ is uniform in s ; it is an \mathcal{L} formula with \mathbf{z} as the only free variables, if any, when s is suppressed. When object arguments of $A(\mathbf{z})$ are instantiated with constants, by Lemma 1, the RHS of each precondition axiom can be translated into a concept in $\mathcal{ALCC}(U)$, when the situation variable s is suppressed.

Successor state axioms \mathcal{D}_{SS} : There is a single SSA for each fluent $F(\mathbf{x}, do(a, s))$. According to the general syntactic form of the SSAs provided in (Reiter 2001), without loss of generality, we can assume that each axiom is as follows:

$$(\forall \mathbf{x}, s, a). F(\mathbf{x}, do(a, s)) \equiv \gamma_F^+(\mathbf{x}, a, s) \vee F(\mathbf{x}, s) \wedge \neg \gamma_F^-(\mathbf{x}, a, s) \quad (2)$$

where each of the γ_F 's are disjunctions either of the form

$$[\exists \mathbf{z}]. a = A(\mathbf{u}) \wedge \phi(x, \mathbf{z}, s), \quad /* \text{ a set of variables } \mathbf{z} \subseteq \mathbf{u}; \text{ may be } \{x\} \in \mathbf{u} */$$

if (2) is a SSA for a dynamic concept $F(x, s)$ with a single object argument x , or

$$[\exists \mathbf{z}]. a = A(\mathbf{u}) \wedge \phi(x, \mathbf{z}, s) \wedge \phi(y, \mathbf{z}, s), \quad /* \mathbf{z} \subseteq \mathbf{u}, \text{ possibly } \{x, y\} \cap \mathbf{u} \neq \emptyset */$$

if (2) is a SSA for a dynamic role $F(x, y, s)$, where $\phi(\mathbf{x}, \mathbf{z}, s)$ is a *context condition* uniform in s saying when an action A can have an effect on the fluent F . The formula $\phi(x, \mathbf{z}, s) \in \mathbf{F}^x$, the formula $\phi(y, \mathbf{z}, s) \in \mathbf{F}^y$, when s is suppressed. A set of variables \mathbf{z} in a context condition $\phi(\mathbf{x}, \mathbf{z}, s)$ must be a subset of object variables \mathbf{u} . If \mathbf{u} in an action function $A(\mathbf{u})$ does not include any \mathbf{z} variable, then there is no $\exists \mathbf{z}$ quantifier.

If not all variables from \mathbf{x} are included in \mathbf{u} , then it is said that $A(\mathbf{u})$ has a *global effect*, since the fluent F experiences changes beyond the objects explicitly named in $A(\mathbf{u})$ (e.g., driving a truck between two locations changes location of *all* boxes loaded in the truck). When a vector of object variables \mathbf{u} contains both \mathbf{x} and \mathbf{z} , we say that the

action $A(\mathbf{u})$ has a *local effect*. A BAT is called a *local-effect BAT* if all of its actions have only local effects. Observe that in a local-effect SSA, when one substitutes a ground action term $A(\mathbf{b}_x, \mathbf{b}_z)$ for a variable a in the formula $[\exists \mathbf{z}].a = A(\mathbf{x}, \mathbf{z}) \wedge \phi(\mathbf{x}, \mathbf{z}, s)$, applying UNA for action terms yields $[\exists \mathbf{z}].\mathbf{x} = \mathbf{b}_x \wedge \mathbf{z} = \mathbf{b}_z \wedge \phi(\mathbf{x}, \mathbf{z}, s)$, and applying $\exists z(z = b \wedge \phi(z)) \equiv \phi(b)$ repeatedly results in the equivalent formula $\mathbf{x} = \mathbf{b}_x \wedge \phi(\mathbf{x}, \mathbf{b}_z, s)$.

Initial Theory \mathcal{D}_{S_0} : The \mathcal{D}_{S_0} is an \mathcal{L} sentence without z variables, i.e., it can be transformed into an $\mathcal{ALCO}(\mathcal{U})$ concept.

A basic action theory \mathcal{D} that satisfies all of the above requirements is called an action theory \mathcal{P} . We note that BATs proposed in [13] are less general than \mathcal{P} , because their axioms should be written using formulas from $FODL$, but $FODL$ is a proper subset of \mathcal{L} . Sometimes, for clarity, when we talk about \mathcal{P} , we say that it is an \mathcal{L} -based BAT, in contrast to $FODL$ -based BATs considered in [13]. The Blocks World is an example of a $FODL$ -based BAT, while Logistics is an example of \mathcal{P} . Logistics cannot be formulated as a $FODL$ -based BAT because it includes actions, e.g., $drive(Truck, Loc_1, Loc_2, City)$, with more than 2 arguments, and the SSA for a dynamic role $At(obj, loc, s)$ uses as a context condition an \mathbf{F}^x formula, while in [13], the SSAs for dynamic roles must be context-free. Subsequently, for brevity, instead of saying that $\phi(s)$ is a SC formula uniform in s that becomes an \mathcal{L} formula when s is suppressed, we say simply that ϕ is an \mathcal{L} formula.

Due to space limitations, we skip introduction to DLs, but the reader can find one in [2]. Recall that the satisfiability problem of a concept and/or the consistency problem of an ABox in the DL language $\mathcal{ALCO}(\mathcal{U})$ can be solved in EXPTIME.

Example 1. As an example of \mathcal{P} , imagine searching for a given file in a depth-first search (DFS) like manner through directories. An action $forward(z_1, z_2, z_3)$ makes forward transition from a current directory z_1 to its child directory z_2 while searching for a file z_3 . It is possible in situation s , if z_2 has never been visited. This is represented using the fluent $vis(z_2, z_3, s)$. A $backtrack(z_1, z_2, z_3)$ transition from z_1 back to its parent z_2 is possible only if all children of z_1 had been visited while searching for a file z_3 . \mathcal{P} also includes situation independent unary predicates $file(x)$, $dir(x)$, and the binary predicate $dirChild(x, y)$ meaning that x is a direct child of y in a file system. The search for a file f in a directory d succeeds when $find(d, f)$ is executed. This action is possible when d actually contains f . This is represented using the fluent $at(d, f, s)$. Using $chmod(z_1, z_2)$ one can toggle in situation s permissions of a directory z_1 between $z_2 = on$ and $z_2 = off$, if the current permission x for this directory z_1 , represented using the fluent $perm(z_1, x, s)$, is such that the values of x and z_2 are opposite. The following are precondition axioms (PA) for all actions (the variables z_i, s are \forall -quantified at front).

$$\begin{aligned}
Poss(forward(z_1, z_2, z_3), s) &\equiv dir(z_1) \wedge dir(z_2) \wedge z_1 \neq z_2 \wedge file(z_3) \wedge \\
&\quad dirChild(z_2, z_1) \wedge \neg vis(z_2, z_3, s) \wedge at(z_1, z_3, s) \\
Poss(backtrack(z_1, z_2, z_3), s) &\equiv dir(z_1) \wedge dir(z_2) \wedge file(z_3) \wedge dirChild(z_1, z_2) \wedge \\
&\quad at(z_1, z_3, s) \wedge \neg \exists y (dirChild(y, z_1) \wedge dir(y) \wedge \neg vis(y, z_3, s)) \\
Poss(find(z_1, z_2), s) &\equiv file(z_1) \wedge dir(z_2) \wedge dirChild(z_1, z_2) \wedge at(z_2, z_1, s) \\
Poss(chmod(z_1, z_2), s) &\equiv dir(z_1) \wedge (z_2 = on \vee z_2 = off) \wedge \\
&\quad \exists x.(perm(z_1, x, s) \wedge x \neq z_2).
\end{aligned}$$

The direct effects of actions are formulated using successor state axioms (SSA). The current DFS for a file y arrives at a directory x when either forward or backtracking transition leads to x ; otherwise, if any other action is executed, it remains at x . Also, the directory x becomes visited as soon as DFS arrives there following some forward transition, but only if the current permission of x is *on* in situation s . Otherwise, forward transition has no effect. Changing permission of a directory x to y has an effect only when DFS for a file is currently located at x in situation s . A file f is found after doing $find(x, z_1$ in a directory z_1 only if permission is *on* for this directory in s .

$$\begin{aligned}
at(x, y, do(a, s)) &\equiv \exists z_1(a = forward(z_1, x, y) \wedge perm(x, on, s)) \vee \\
&\quad \exists z_1(a = backtrack(z_1, x, y)) \vee \\
at(x, y, s) \wedge \neg \exists z_1(a = forward(x, z_1, y) \wedge perm(z_1, on, s)) \wedge \\
&\quad \neg \exists z_1(a = backtrack(x, z_1, y)) \\
vis(x, y, do(a, s)) &\equiv \exists z_1(a = forward(z_1, x, y) \wedge perm(x, on, s)) \vee vis(x, y, s) \\
perm(x, y, do(a, s)) &\equiv a = chmod(x, y) \wedge \exists y(at(x, y, s) \wedge y = y) \vee \\
&\quad perm(x, y, s) \wedge \neg \exists z_1(a = chmod(x, z_1) \wedge y \neq z_1 \wedge \exists y.at(x, y, s) \wedge y = y) \\
found(x, do(a, s)) &\equiv \exists z_1(a = find(x, z_1) \wedge perm(z_1, on, s)) \vee found(x, s).
\end{aligned}$$

These SSAs satisfy syntactic constraints in \mathcal{P} , but they cannot be formulated as FO_{DL} -based SSAs considered in [13] since SSAs for the dynamic roles *at* and *perm* have context conditions and mention action functions with more than 2 arguments. Clearly, neither PAs, nor SSAs can be translated to a DL, but nevertheless, there are instances of the projection problem in this BAT that can be reduced to SAT in a DL.

The Projection Problem in \mathcal{P}

Let \mathcal{D} be a description logic based BAT defined in [13], $\alpha_1, \dots, \alpha_n$ be a sequence of ground action terms, and $Goal(s)$ be a query formula uniform in s such that it can be transformed into an $\mathcal{ALCO}(\mathcal{U})$ concept, if s is suppressed. Subsequently, we call a query $Goal(S)$ a *regressible formula*, if S is a ground situation term. One of the most important reasoning tasks in the SC is the *projection problem*, that is, to determine whether $\mathcal{D} \models Goal(do([\alpha_1, \dots, \alpha_n], S_0))$. Another basic reasoning task is the *executability problem*: whether all ground actions in $\alpha_1, \dots, \alpha_n$ can be consecutively executed. This can be reduced to the projection problem using the precondition axioms, and for this reason we no longer consider it. Planning and high-level program execution are two important settings where the executability and projection problems arise naturally. *Regression* is a central computational mechanism that forms the basis of automated solutions to the executability and projection tasks in the SC [22]. A recursive definition of the *modified regression operator* \mathcal{R} on any regressible formula $Goal(S)$ is given in [13]. The modified regression operator makes sure that the only two available object variables x, y are re-used when regressing a quantified formula in contrast to Reiter's regression, where new variables are introduced. For a regressible formula $Goal(S)$, we use notation $\mathcal{R}[Goal(S)]$ to denote the regressed formula uniform in S_0 that results from replacing repeatedly fluent atoms about $do(\alpha, s)$ by logically equivalent expressions about s as given by the RHS of SSAs, until such replacements no longer can be made; this is why the regressed formula is uniform in S_0 . For any static concept $C(x)$ and role $R(x, y)$, by definition of regression $\mathcal{R}[C(x)] = C(x)$ and $\mathcal{R}[R(x, y)] = R(x, y)$.

The regression theorem (Theorem 8) proved in [13] shows that $\mathcal{R}[Goal(S)]$ is a FO_{DL} formula, when S_0 is suppressed and, as a consequence, one can reduce the projection problem for a regressable sentence $Goal(S)$ to the satisfiability problem in $\mathcal{ALCO}(U)$ as long as a BAT \mathcal{D} satisfies syntactic restrictions due to using FO_{DL} formulas in axioms:

$$\mathcal{D} \models Goal \text{ iff } \mathcal{D}_{S_0} \models \mathcal{R}[Goal(S)],$$

where it is assumed that \mathcal{D}_{S_0} includes UNA , unique name axioms for objects. (Unique name axioms for actions are used by modified regression, and they are no longer required when regression terminates.) This statement is proved in [13] for an extended BAT that additionally includes a set of axioms $\mathcal{D}_T = \mathcal{D}_{T,st} \cup \mathcal{D}_{T,dyn}$, where the static TBox $\mathcal{D}_{T,st}$ is an acyclic set of *concept definitions* that mentions only situation independent predicates (in [13], \mathcal{D}_{S_0} includes $\mathcal{D}_{T,st}$), while dynamic TBox $\mathcal{D}_{T,dyn}$ is an acyclic set of definitions such that it has occurrences of fluents, but defined fluents are mentioned only in the RHS of SSAs, and they are eliminated by the modified regression operator using *lazy unfolding*. For example, $\mathcal{D}_{T,st}$ may include situation independent static definitions such as “vehicle is a truck or an airplane”, while $\mathcal{D}_{T,dyn}$ may include convenient situation dependent abbreviations like $Movable(x, s) \equiv Loaded(x, s) \wedge \exists y In(x, y, s)$. The previously mentioned acyclicity assumption originates in [4].

We would like to eliminate a previous assumption that $\mathcal{D}_{T,st}$ is acyclic. For simplicity, let us consider a case when $\mathcal{D}_{T,dyn} = \emptyset$. Let \mathcal{D} be \mathcal{P} such that its initial theory \mathcal{D}_{S_0} is augmented with an arbitrary satisfiable static TBox $\mathcal{D}_{T,st}$ that may include *general concept inclusions* between $\mathcal{ALCO}(U)$ concepts. (This TBox can be expressed as an $\mathcal{ALCO}(U)$ concept.) Then, by the relative satisfiability theorem from [21], $\Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ is satisfiable iff $UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ is satisfiable, i.e., the presence of a static satisfiable ontology is harmless. Moreover, since regression does not affect the predicates without a situation term, in other words, since axioms in $\mathcal{D}_{T,st}$ are invariant wrt the regression operator, it can be used to answer “static” queries and to reduce the projection problem to the satisfiability in $\mathcal{ALCO}(U)$: $\Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st} \models Goal$ iff $UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st} \models Goal$, when $Goal$ is an \mathcal{L} sentence without z -variables that has no occurrences of fluents (a “static” query), and UNA includes unique name axioms only for objects. This simple observation is a consequence of Lemma 1 and the regression theorem from [21]. In addition, in \mathcal{P} we can prove that formulas from \mathcal{L} remain to be in \mathcal{L} after regression.

Theorem 1. *Let \mathcal{D} be an \mathcal{L} -based BAT (a theory \mathcal{P}), ϕ be a regressable \mathcal{L} formula, and α a ground action. The result of regressing $\phi[(do(\alpha, S_0))]$, denoted by $\mathcal{R}[\phi(do(\alpha, S_0))]$, is a formula uniform in situation S_0 that is an \mathcal{L} -formula if S_0 is suppressed.*

This can be proved similarly to Lemma 2 from Section 5.4 in [13] that is proved for a FO_{DL} -based BAT. However, this does not follow directly from [13, 12] because SSA for dynamic roles may have context conditions in \mathcal{P} , but in [13, 12] it was assumed that SSA for dynamic roles are context free. Also, recall that FO_{DL} is a proper subset of \mathcal{L} . The proof is long and laborious because regression is a syntactic operation, and the SSAs in \mathcal{P} may have several different syntactic forms, but we have to show that if we start with a DL-like formula, then after a single step of regression we get a formula that remains DL-like. As a consequence, for the “dynamic” queries, we have the following.

Theorem 2. Let $\mathcal{D} = \Sigma \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup \text{UNA} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ be \mathcal{P} augmented with a (static) general $\mathcal{ALCO}(\mathcal{U})$ TBox, $\phi(S)$ be a regressive z -free \mathcal{L} sentence, and S be a ground situation. Then the projection problem can be reduced to satisfiability in $\mathcal{ALCO}(\mathcal{U})$:

$$\begin{aligned} \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup \text{UNA} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st} \models \phi(S) \quad \text{iff} \\ \text{UNA} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st} \models \mathcal{R}[\phi(S)] \end{aligned}$$

This follows from Theorem 1 by induction on the length of the situation term S , from Lemma 1, and from the fact that $\text{UNA} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ can be transformed into an $\mathcal{ALCO}(\mathcal{U})$ concept. This theorem is important because it shows that any static $\mathcal{ALCO}(\mathcal{U})$ ontology can be seamlessly integrated with reasoning about actions in \mathcal{P} . Also, one can add an acyclic dynamic TBox $\mathcal{D}_{T,dyn}$ to \mathcal{P} without any difficulties, as in [13]. However, [4, 17, 6] and others argue that a general dynamic TBox leads to serious difficulties. While [4] does not consider a general static TBox $\mathcal{D}_{T,st}$, it could be added, e.g., by internalizing $\mathcal{D}_{T,st}$ into an $\mathcal{ALCO}(\mathcal{U})$ concept and including it as an ABox assertion wrt a dummy individual. This trick was not considered in [4], because the universal role U is required for this trick to work, but U was missing in [4].

Example 1 (cont.) We would like to adapt for our purposes an example of a general TBox from the paper by Giuseppe De Giacomo, Maurizio Lenzerini ‘‘TBox and ABox Reasoning in Expressive Description Logics’’, KR 1996, pages 316-327). Suppose that a static TBox has the following general concept inclusions:

$$\begin{aligned} \text{dir} &\sqsubseteq \forall \text{dirChild}^-. (\text{dir} \sqcup \text{file}) \sqcap \leq 1 \text{dirChild.dir} \\ \text{file} &\sqsubseteq \neg \text{dir} \sqcap \forall \text{dirChild}^-. \perp \end{aligned}$$

Let an initial \mathcal{D}_{S_0} be the following theory (written as \mathcal{L} formula for brevity):

$$\begin{aligned} &\text{dir}(\text{home}) \wedge \text{dir}(\text{mes}) \wedge \text{dir}(\text{root}) \wedge \text{dir}(\text{wyehia}) \wedge \text{file}(f1) \wedge \text{file}(f2) \\ &\text{dirChild}(f1, \text{mes}) \wedge \text{dirChild}(f2, \text{wyehia}) \wedge \text{dirChild}(\text{home}, \text{root}) \wedge \\ &\text{dirChild}(\text{mes}, \text{home}) \wedge \text{dirChild}(\text{wyehia}, \text{home}) \wedge \\ &\text{at}(\text{wyehia}, f1, S_0) \wedge \forall x. (\neg(\text{dir}(x) \vee \text{file}(x)) \vee \text{perm}(x, \text{on}, S_0)) \end{aligned}$$

The UNA for object constants (represented as nominals in $\mathcal{ALCO}(\mathcal{U})$):

$$\{f1\} \neq \{f2\} \neq \{\text{home}\} \neq \{\text{mes}\} \neq \{\text{off}\} \neq \{\text{on}\} \neq \{\text{root}\} \neq \{\text{wyehia}\}$$

Let the projection query be whether $\mathcal{D} \cup \text{TBox} \models \text{found}(f1, S)$, where S is $\text{do}([\text{backtrack}(\text{wyehia}, \text{home}, f1), \text{forward}(\text{home}, \text{mes}, f1), \text{find}(f1, \text{mes})], S_0)$. Then, it is easy to see that the regressed query is

$$((f1 = f1 \wedge \text{perm}(\text{mes}, \text{on}, S_0)) \vee \text{found}(f1, S_0))$$

This example demonstrates that we managed to solve the projection problem in the presence of a general expressive static TBox. This example has been implemented: axioms are implemented in XML and regression of a query was computed using a C++ program, see details at <http://www.scs.ryerson.ca/mes/dl2012.zip>

Progression in \mathcal{P}

In this section, we use the notion of forgetting about a sequence of ground atoms, the notion of progression in SC, the fact about definability of progression in FOL for local effect BATs, and notation introduced in [15, 16, 18]. Recall that \mathcal{P} is any \mathcal{L} -based BAT. It is easy to give an example of \mathcal{P} with global effect actions such that progression of \mathcal{D}_{S_0} is not definable as a z -free \mathcal{L} sentence. Subsequently, we consider only local-effect \mathcal{P} action theories, and we talk about z -free \mathcal{L} sentences that can be transformed into

an $\mathcal{ALCO}(\mathcal{U})$ concept. Below, we prove that progression of a z -free \mathcal{L} sentence \mathcal{D}_{S_0} is still expressible as a z -free \mathcal{L} sentence \mathcal{D}_{S_α} (here and subsequently, for brevity, we talk about situation-suppressed sentences). This does not follow from Theorem 3.6 in [18] about definability of progression in FOL for local-effect BATs, since our initial theory \mathcal{D}_{S_0} is formulated in a strict subset of FO^2 language, and it is not obvious at all whether in \mathcal{P} progression \mathcal{D}_{S_α} of \mathcal{D}_{S_0} can still be defined within our language. Since progression involves forgetting about old values of fluents and computing new values, we need a couple of intermediate lemmas. First, we show that new fluent values can be expressed in \mathcal{L} . Then, we prove that the result of forgetting about ground fluents in \mathcal{D}_{S_0} affected by a ground action α remains to be a z -free \mathcal{L} sentence.

Lemma 2. *Let \mathcal{D} be a local effect \mathcal{P} , α a ground action, and $\Omega(S_0)$ be the characteristic set of α with respect to \mathcal{D} . Then $\mathcal{D}_{SS}[\Omega]$ is a set of \mathcal{L} sentences without occurrences of z -variables, when the situation terms are suppressed.*

The characteristic set $\Omega(S_0)$ is a set of ground fluents affected by α . Because they change values, we have to forget their old values. To compute new values for them, we instantiate \mathcal{D}_{SS} w.r.t. $\Omega(S_0)$, do simplification and obtain the set of sentences $F(\mathbf{t}, S_\alpha) \equiv \Phi_F(\mathbf{t}, \alpha, S_0)$, which are denoted as $\mathcal{D}_{SS}[\Omega]$, where $S_\alpha = do(\alpha, S_0)$, and $\Phi_F(\mathbf{t}, \alpha, S_0)$ is a z -free \mathcal{L} sentence representing the RHS of a SSA for the fluent F . $F(\mathbf{t}, S_\alpha)$ and $\Phi_F(\mathbf{t}, \alpha, S_0)$ mention different situation terms. However, $F(\mathbf{t}, S_\alpha)$ can never occur in Φ_F or any RHS of SSA of other fluents because they are all uniform in S_0 . Also, none of the ground fluents to be subsequently forgotten are relevant to $F(\mathbf{t}, S_\alpha)$ simply because it is the value of F in a different situation. Consequently, we can replace $F(\mathbf{t}, S_\alpha)$ temporarily by some atom F_t until forgetting of $\Omega(S_0)$ is completed, and then put it back while preserving logical equivalence. The next lemma shows that forgetting about ground atoms $\Omega(S_0)$ in an \mathcal{L} formula results in an \mathcal{L} formula.

Lemma 3. *Let ϕ be a \mathbf{F}^x (or \mathbf{F}^y) formula and θ a truth assignment to some of the atoms $P(\mathbf{t}_j)$ occurring in this formula (if any), then $\phi[\theta]$ remains a \mathbf{F}^x (\mathbf{F}^y) formula.*

Notation $\phi[\theta]$ for forgetting about several ground atoms, introduced in [18], means the result of replacing every occurrence of an atom $P(\mathbf{x})$ in ϕ by $\bigvee_{j=1}^m (\mathbf{x} = \mathbf{t}_j \wedge \theta[P(\mathbf{t}_j)]) \vee (\bigwedge_{j=1}^m \mathbf{x} \neq \mathbf{t}_j) \wedge P(\mathbf{x})$. This Lemma is proved by induction over structure of ϕ .

Theorem 3. *Let \mathcal{D} be a local-effect BAT based on \mathcal{L} and α a ground action. Let $\Omega(s)$ be the characteristic set of α . Then the following formula is a progression of \mathcal{D}_{S_0} w.r.t. α and this formula is an \mathcal{L} sentence:*

$$\bigwedge UNA \wedge \bigvee_{\theta \in \mathcal{M}(\Omega(S_0))} (\bigwedge \mathcal{D}_{S_0} \wedge \bigwedge \mathcal{D}_{SS}[\Omega][\theta] (S_\alpha/S_0)) \quad (3)$$

Proof: This is a consequence of Lemmas (2), (3) and Theorem 3.6 from [18]. Note that the final formula is uniform in S_α . This theorem is important for our work because it shows for \mathcal{P} that if an initial theory \mathcal{D}_{S_0} is expressible as an $\mathcal{ALCO}(\mathcal{U})$ -like concept, then progression \mathcal{D}_{S_α} is also expressible as an $\mathcal{ALCO}(\mathcal{U})$ -like concept.

Theorem 3 shows progression \mathcal{D}_{S_α} can be translated to $\mathcal{ALCO}(\mathcal{U})$, but in a general case, the size of progression can be much larger than the size of \mathcal{D}_{S_0} . If one wants to solve the projection problem by computing progression for a sequence of action, then one has to find special cases of an initial theory \mathcal{D}_{S_0} such that the size of progression remains linear w.r.t. the size of \mathcal{D}_{S_0} . It turns out that progression is computationally

tractable if an initial \mathcal{D}_{S_0} is in *proper*⁺ form [18], where *proper*⁺ theories generalize databases by allowing incomplete disjunctive knowledge about some of the named elements of the domain [14]. A *proper*⁺ knowledge base (KB) is more general than a *proper* KB, which is equivalent to a possibly infinite consistent set of ground literals. We show that in \mathcal{P} , if \mathcal{D}_{S_0} is a set of *proper*⁺ formulas that can be translated into $\mathcal{ALCO}(\mathcal{U})$, i.e., a boolean \mathcal{ALC} ABox, then progression of \mathcal{D}_{S_0} in our normal form can be computed efficiently, and the normal form can be maintained without introducing any new variables. To achieve this, we introduce a new p^+ normal form. We show that a KB in our p^+ normal form can be equivalently transformed into the same normal form after forgetting about old values of fluents, and none of the intermediate logical transformations require introducing new variables to preserve logical equivalence. The fact that forgetting in our normal form KB can be accomplished without introducing new variables is novelty that does not follow from [18]. Also, we show that after progression the size of the progressed KB is linear wrt to the size of the initial KB, i.e., progression can be computed efficiently. Once an initial theory \mathcal{D}_{S_0} has been progressed to \mathcal{D}_{S_a} , solving the projection problem can be done using any $\mathcal{ALCO}(\mathcal{U})$ satisfiability solver.

Due to lack of space we omit all technical details, but the interested readers can find them in the longer version at <http://www.scs.ryerson.ca/mes/dl2012.zip>

Discussion and Future Work

Main contributions of our paper are as follows. First, we define a logical theory \mathcal{P} integrating reasoning about action with DLs such that \mathcal{P} is more expressive than theories from [12, 13]. Second, Theorem 2 (regression in \mathcal{P}) shouldn't be underestimated. It shows existing ontologies (with a general $\mathcal{ALCO}(\mathcal{U})$ static TBox) can be seamlessly integrated with \mathcal{P} when solving the projection problem. To the best of our knowledge, this seamless integration of DLs and reasoning about actions has never been proposed before. For example, [4, 13] allowed only acyclic dynamic TBox (that can be easily added to \mathcal{P} too). Third, Theorem 3 is a new non-trivial statement that doesn't follow from [18]. It is important because it guarantees that progression of $\mathcal{ALCO}(\mathcal{U})$ KBs can still be formulated in the same language, and consequently, one can continue computing progression for subsequent actions. Fourth, theorems (not included in this version) about maintaining our new p^+ normal form after forgetting are proved using new techniques. They don't follow from [18], where progression was studied in FOL.

An approach to integrating DLs and reasoning about actions proposed in [4] inspired a number of subsequent papers including [13], where the reader can find extensive comparison and discussion. The approach proposed in [4] is expressive, and it can be used to represent many popular AI action theories. However, one can answer only ground projection queries using their approach, but Theorem 2 shows we can use regression to answer projection queries with quantifiers over object arguments in fluents. Also, our regression can be used to solve the projection problem in a BAT where some actions have global effects, but the approach proposed in [4] can answer projection queries only in local effect BATs. In any case, it is important to compare our implementations with an implementation based on [4] for the common classes of queries and theories. The first step in this direction is taken in [3]. Due to lack of space we couldn't discuss other related work, but all related publications are very extensively discussed in [4, 6, 13, 17].

References

1. Artale, A., Franconi, E.: A survey of temporal extensions of description logics. *Ann. Math. Artif. Intell.* 30(1-4), 171–210 (2000)
2. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) *Handbook of Knowledge Representation*, pp. 135–179. Elsevier (2007)
3. Baader, F., Lippmann, M., Liu, H., Soutchanski, M., Yehia, W.: Experimental results on solving the projection problem in action formalisms based on description logics. In: *Proc. of the 25th Intern. Workshop on Description Logics* (2012)
4. Baader, F., Lutz, C., Miličić, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: *Proceedings of the 20th AAAI Conference*. pp. 572–577. Pittsburgh, PA, USA (2005), extended version is available as LTCS-Report-05-02 at <http://lat.inf.tu-dresden.de/research/reports.html>
5. Badea, L.: Planning in description logics: Deduction versus satisfiability testing. In: *Proc. of the Intern. Workshop on Description Logics* (1998)
6. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Actions and programs over description logic ontologies. In: *Proc. of the Intern. Workshop on Description Logics* (2007)
7. Chang, L., Shi, Z., Qiu, L., Lin, F.: Dynamic description logic: Embracing actions into description logic. In: *Proc. of the Intern. Workshop on Description Logics* (2007)
8. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. *J. Web Sem.* 6(4), 309–322 (2008)
9. De Giacomo, G., Iocchi, L., Nardi, D., Rosati, R.: A theory and implementation of cognitive mobile robots. *J. Log. Comput.* 9(5), 759–785 (1999)
10. De Giacomo, G., Lenzerini, M.: PDL-based framework for reasoning about actions. In: Gori, M., Soda, G. (eds.) *AI*IA. Lecture Notes in Computer Science*, vol. 992, pp. 103–114. Springer (1995)
11. Devanbu, P.T., Litman, D.J.: Taxonomic plan reasoning. *Artif. Intell.* 84(1-2), 1–35 (1996)
12. Gu, Y.: *Advanced Reasoning about Dynamical Systems*. Ph.D. thesis, Department of Computer Science, University of Toronto, Canada (2010)
13. Gu, Y., Soutchanski, M.: A description logic based situation calculus. *Ann. Math. Artif. Intell.* 58(1-2), 3–83 (2010)
14. Lakemeyer, G., Levesque, H.J.: Evaluation-based reasoning with disjunctive information in first-order knowledge bases. In: *Proc. of KR-02*. pp. 73–81 (2002)
15. Lin, F., Reiter, R.: Forget it! In: *Proceedings of the AAAI Fall Symposium on Relevance*. pp. 154–159 (1994)
16. Lin, F., Reiter, R.: How to progress a database. *Artificial Intelligence* 92, 131–167 (1997)
17. Liu, H., Lutz, C., Miličić, M., Wolter, F.: Reasoning about actions using description logics with general TBoxes. In: *Logics in Artificial Intelligence, Lecture Notes in Computer Science*, vol. 4160, pp. 266–279. Springer Berlin / Heidelberg (2006)
18. Liu, Y., Lakemeyer, G.: On first-order definability and computability of progression for local-effect actions and beyond. In: Boutilier, C. (ed.) *IJCAI*. pp. 860–866 (2009)
19. Lutz, C., Sattler, U.: A proposal for describing services with dls. In: *Proc. of the 15th Intern. Workshop on Description Logics* (2002)
20. McDermott, D.V.: The 1998 AI planning systems competition. *AI Magazine* 21(2), 35–55 (2000)
21. Pirri, F., Reiter, R.: Some contributions to the metatheory of the situation calculus. *Journal of the ACM* 46(3), 325–364 (1999)
22. Reiter, R.: *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. The MIT Press (2001)
23. Wolter, F., Zakharyashev, M.: Dynamic description logics. In: *Advances in Modal Logic*. pp. 431–446. CSLI Publications (1998)

Concurrent Classification of OWL Ontologies – An Empirical Evaluation

Mina Aslani and Volker Haarslev

Concordia University, Montreal, Quebec, Canada

Abstract. This paper describes our progress in developing algorithms for concurrent classification of OWL ontologies. We refactored the architecture of our research prototype and its employed algorithms by integrating lock-free data structures and adopting various optimizations to reduce overhead. In comparison to our earlier work we increased the size of classified ontologies by one order of magnitude, i.e., the size of processed ontologies is now beyond a quarter million of OWL classes. The main focus of this paper is an empirical evaluation with huge ontologies that demonstrates an excellent speedup that almost directly corresponds to the number of used processors or cores.

1 Introduction

Parallel algorithms for description logic (DL) reasoning were first explored in the FLEX system [3] where various distributed message-passing schemes for rule execution were evaluated. The reported results seemed to be promising but the research suffered from severe limitations due to the hardware available for experiments at that time. The only other work on parallelizing DL reasoning [9] reported promising results using multi-core and multi-processor hardware, where the parallel treatment of disjunctions and individual merging (due to number restrictions) is explored. In [11] an approach on distributed reasoning for *ALCHIQ* is presented that is based on resolution techniques but does not address optimizations for TBox classification.

Other work has studied concurrency in light-weight ontology languages. There is a distributed Map Reduce approach algorithm for \mathcal{EL}^+ , however no experiments had been reported on the proposed algorithms [10]. Other work focuses on distributed reasoning, and these approaches are different than ours as they manage large-scale data which is beyond the memory of a single machine [11, 14, 6, 8, 12]. There also exists work on parallel distributed RDF inferencing (e.g., [13]) and parallel reasoning in first-order theorem proving but due to completely different proof techniques (resolution versus tableaux) and reasoning architectures this is not considered as relevant here. Another work presents an optimized consequence-based procedure for classification of ontologies but it only addresses the DL \mathcal{EL} [7].

The work in this paper is an extension of our work on Parallel TBox classification [1]. Compared to our previous work, this paper reports on an enhanced lock-free version of algorithms utilizing concurrency in a multi-core environment, optimizations that increase the performance, a performance evaluation with huge real-world ontologies in the range of 300K OWL classes (DL concepts) such as SNOMED. Our prototype not only addresses huge real-world ontologies but also does not compromise on DL complexity. It can process much more complex DLs (e.g., at least *SHIQ*) than \mathcal{EL} , and

provides an excellent speedup considering that no particular DL related optimization technique is used. The implemented prototype system performs concurrent TBox classification based on various parameters such as number of threads, size of partitions assigned to threads, and number of processors. Our evaluation demonstrates impressive performance improvements where the number of available processors almost linearly decreases the processing time due to a small overhead. It is important to note that the focus of this research is on exploring algorithms for concurrent TBox classification and not on developing a highly optimized DL reasoner. We are currently only interested in the speedup factor obtained from comparing sequential and parallel runs of our prototype.

2 The Concurrent TBox Classifier

This section describes the architecture of the implemented system and its underlying *sound and complete* algorithm for concurrent classification of DL ontologies. To compute the hierarchy in parallel, we developed a Java application using a multi-threaded architecture providing control parameters such as number of threads, number of concepts (also called partition size) to be inserted per thread, and number of processors. As thoroughly explained in [1], the program reads an input file containing a list of concept names to be classified and information about them which is generated by the OWL reasoner Racer [4]. Racer is only used for generating the input files for our prototype. The per-concept information available in the file includes the concept name, its parents (in the complete taxonomy), so-called told subsumers and disjoints, and pseudo model [5] information. This architecture was deliberately designed to facilitate our experiments by using existing OWL reasoners to generate auxiliary information and to make the Concurrent TBox Classifier independent of particular DLs.

The preprocessing algorithm uses a topological sorting similar to [1] and the order for processing concepts is based on the topologically sorted list. To manage concurrency and multi-threading in our system, as described in [1], a single-shared global tree approach is used. Also, to classify the TBox, two symmetric tasks are employed, i.e., the so-called enhanced tree traversal method [2] using top (bottom) search to compute the parents (children) of a concept to be inserted into the taxonomy.

In [1], we first introduced our algorithms for parallel classification and reported considerable performance improvements but we could only process relatively small ontologies. In this paper, we introduce the enhanced concurrent version of these algorithms, i.e., Algorithms 2, 6 and 7. In order to make the paper self-contained we repeat Algorithms 1, 3, 4 and 5 from [1].

The procedure `parallel_tbox_classification` is sketched in Algorithm 1. It is called with a list of named concepts and sorts them in topological order with respect to the initial taxonomy created from already known told ancestors and descendants of each concept (using the told subsumer information). The classifier assigns in a round-robin manner partitions with a fixed size from the concept list to idle threads and activates these threads with their assigned partition using the procedure `insert_partition` outlined in Algorithm 2. All threads work in parallel with the goal to construct a global subsumption tree (taxonomy). They also share a global array *located_concepts* indexed by thread

Algorithm 1 `parallel_tbox_classification(concept_list)`

topological_order_list \leftarrow `topological_order(concept_list)`

repeat

wait until an idle thread t_i becomes available

select a partition p_i from *topological_order_list*

run thread t_i with `insert_partition(p_i, t_i)`

until all concepts in *topological_order_list* are inserted

identifications. Using the Concurrency package in Java, synchronization on the nodes of the global tree as well as the entries in the global array have now been eliminated.

The procedure `insert_partition` inserts all concepts of a given partition into the global taxonomy. We use Concurrent collections from the `java.util.concurrent` package. This package supplies Collection implementations which are thread-safe and designed for use in multi-threaded contexts. Therefore, for updating a concept or its parents or children, no locking mechanism for the affected nodes of the global tree is needed anymore. Algorithm 2 first performs for each concept *new* the top-search phase (starting from the top concept (\top)) and possibly repeats the top-search phase for *new* if other threads updated the list of children of its parents. Then, it sets the parents of *new*. Afterwards the bottom-search phase (starting from the bottom concept (\perp)) is performed. Analogously to the top-search phase, the bottom search is possibly repeated and sets the children of *new*. After finishing the top and bottom search for *new*, the node *new* is added to the entries in *located_concepts* of all other busy threads; it is also checked whether other threads updated the entry in *located_concepts* for this thread. If this was the case, the top and/or bottom search need to be repeated correspondingly.

To reduce overhead in re-running of top or bottom search, we only re-run twice. If the concept *new* is still not ready to be inserted; e.g., there is any interaction between *new* and a concept in *located_concepts*; it will be added to the partition list of concepts (to be located later), and also eliminated from the other busy threads' *located_concepts* list, otherwise, *new* can be inserted into the taxonomy using Algorithm 7. In order to avoid unnecessary tree traversals and tableau subsumption tests when computing the subsumption hierarchy, the parallel classifier adapted the enhanced traversal method [2], which is an algorithm that was designed for sequential execution. Algorithms 3 and 4¹ outline the traversal procedures for the top-search phase.

The possible incompleteness caused by parallel classification [1] can be characterized by the following two scenarios: *Scenario I*: In top search, as the new concept is pushed downward, right after the children of the current concept have been processed, at least one new child is added by another thread. In this scenario, the top search for the concept *new* is not aware of the recent change and this might cause missing subsumptions if there is any interaction between the concept *new* and the added children. The same might happen in bottom search if the bottom search for the concept *new* is not informed of the recent change to the list of parents of the current node. *Scenario II*: Between the time that top search has been started to find the location of the concept *new* in the taxonomy and the time that its location has been decided, another thread has

¹ Algorithm `found_in_ancestors(current, new)` checks if *current* is an ancestor of *new*.

Algorithm 2 *insert_partition(partition, id)*

```
for all new  $\in$  partition do
  rerun  $\leftarrow$  0
  finish_rerun  $\leftarrow$  false
  parents  $\leftarrow$  top_search(new,  $\top$ )
  while  $\neg$  consistent_in_top_search(parents, new) do
    parents  $\leftarrow$  top_search(new,  $\top$ )
  predecessors(new)  $\leftarrow$  parents
  children  $\leftarrow$  bottom_search(new,  $\perp$ )
  while  $\neg$  consistent_in_bottom_search(children, new) do
    children  $\leftarrow$  bottom_search(new,  $\perp$ )
  successors(new)  $\leftarrow$  children
  for all busy threads  $t_i \neq id$  do
    located_concepts( $t_i$ )  $\leftarrow$  located_concepts( $t_i$ )  $\cup$  {new}
    check  $\leftarrow$  check_if_concept_has_interaction(new, located_concepts(id))
  while (check  $\neq$  0) and  $\neg$ finish_rerun do
    if rerun < 3 then
      if check = 1 then
        new_predecessors  $\leftarrow$  top_search(new,  $\top$ )
        rerun  $\leftarrow$  rerun + 1
        predecessors(new)  $\leftarrow$  new_predecessors
      if check = 2 then
        new_successors  $\leftarrow$  bottom_search(new,  $\perp$ )
        rerun  $\leftarrow$  rerun + 1
        successors(new)  $\leftarrow$  new_successors
      check  $\leftarrow$  check_if_concept_has_interaction(new, located_concepts(id))
    else
      finish_rerun  $\leftarrow$  true
      for all busy threads  $t_i \neq id$  do
        located_concepts( $t_i$ )  $\leftarrow$  located_concepts( $t_i$ )  $\setminus$  {new}
    if  $\neg$ finish_rerun then
      insert_concept_in_tbox(new, predecessors(new), successors(new))
```

placed at least one concept into the hierarchy which the concept *new* has an interaction with. Again, this might cause missing subsumptions and is analogously also applicable to bottom search.

Both scenarios are properly addressed in Algorithm 2 to ensure completeness. Every time a thread locates a concept in the taxonomy, it notifies the other threads by adding this concept name to their “located_concepts” list. Therefore, as soon as a thread finds the parents and children of the concept *new* by running *top_search* and *bottom_search*; it checks if there is any interaction between concept *new* and the concepts located in the “located_concepts” list. Based on the interaction, *top_search* or *bottom_search* needs to be repeated accordingly. If no possible situations for incompleteness are discovered anymore, Algorithm 7 is called. To resolve the possible incompleteness we utilize Algo-

Algorithm 3 $\text{top_search}(new, current)$

```
mark(current, 'visited')
pos-succ  $\leftarrow \emptyset$ 
captured_successors(new)(current)  $\leftarrow$  successors(current)
for all  $y \in$  successors(current) do
  if enhanced_top_subs( $y, new$ ) then
    pos-succ  $\leftarrow$  pos-succ  $\cup \{y\}$ 
if pos-succ =  $\emptyset$  then
  return {current}
else
  result  $\leftarrow \emptyset$ 
  for all  $y \in$  pos-succ do
    if  $y$  not marked as 'visited' then
      result  $\leftarrow$  result  $\cup$  top_search(new,  $y$ )
  return result
```

Algorithm 4 $\text{enhanced_top_subs}(current, new)$

```
if current marked as 'positive' then
  return true
else if current marked as 'negative' then
  return false
else if for all  $z \in$  predecessors(current)
  enhanced_top_subs( $z, new$ )
  and found_in_ancestors(current, new) then
  mark(current, 'positive')
  return true
else
  mark(current, 'negative')
  return false
```

gorithms 5 and 6.² The procedure `consistent_in_bottom_search` is not shown here because it mirrors `consistent_in_top_search`.

3 Evaluation

In the previous section, we explained the algorithms used in our Concurrent TBox Classifier. In this section, we study the scalability and performance of our prototype. Here, we would like to explain the behavior of our system when we run it in a (i) sequential or (ii) parallel multi-processor environment. We also describe how the prototype performs when we have huge real-world ontologies with different DL complexities. Therefore, in the remaining of this section, we report on the conducted experiments.

We first provide a description of the used platform and the implemented prototype, then we describe the test cases used to evaluate Concurrent TBox Classifier and provide

² Algorithm `interaction_possible(new, concept)` uses pseudo model merging [5] to decide whether a subsumption is possible between *new* and *concept*.

Algorithm 5 consistent_in_top_search(*parents,new*)

```
for all pred ∈ parents do  
  if successors(pred) ≠ captured_successors(new)(pred) then  
    diff ← successors(pred) \ captured_successors(new)(pred)  
    for all child ∈ diff do  
      if found_in_ancestors(child,new) then  
        return false  
return true
```

Algorithm 6 check_if_concept_has_interaction(*new,located_concepts*)

The return value indicates whether and what type of re-run needs to be done:
0 : No re-run in needed
1 : Re-run TopSearch because a possible parent could have been overlooked
2 : Re-run BottomSearch because a possible child could have been overlooked

```
if located_concepts = ∅ then  
  return 0  
else  
  for all concept ∈ located_concepts do  
    if interaction_possible(new,concept) then  
      if found_in_ancestors(new,concept) then  
        return 2  
      else  
        return 1  
    else if interaction_possible(concept,new) then  
      if found_in_ancestors(new,concept) then  
        return 2  
      else  
        return 1  
return 0
```

an overview of the parameters used in the experiments. Finally, we show the results and discuss the performance of the classifier. In addition, the measured runtimes in the figures are shown in seconds using a logarithmic scale.

Platform and implementation All the experiments were conducted on a high performance parallel computing cluster. The nodes in the cluster run an HP-version of RedHat Enterprise Linux for 64 bit processors, with HP’s own XC cluster software stack. To evaluate our approach, Concurrent TBox Classifier has been implemented in Java using lock-free data structures from the java.util.concurrent package with minimal synchronization.

Test cases Table 1 shows a collection of 9 mostly publicly available real-world ontologies. Note that the chosen test cases exhibit different sizes, structure, and DL complexities. The benchmark ontologies are characterized by their name, size in number of named concepts or classes, and used DL.

Parameters used in experiments The parameters used in our empirical evaluation and their meaning are described below (the default parameter value in shown in bold).

Algorithm 7 insert_concept_in_tbox(*new*, *predecessors*, *successors*)

for all *pred* \in *predecessors* **do**
 successors(pred) \leftarrow *successors(pred)* \cup {*new*}
for all *succ* \in *successors* **do**
 predecessors(succ) \leftarrow *predecessors(succ)* \cup {*new*}

Table 1. Characteristics of the used test ontologies (e.g., \mathcal{LH} denotes the DL allowing only conjunction and role hierarchies, and unfoldable TBoxes)

Ontology	DL language	No. of named concepts
Embassi-2	\mathcal{ALCHN}	657
Galen1	\mathcal{ALCH}	2,730
LargeTestOntology	$\mathcal{ELHR}+$	5,584
Tambis-2a	\mathcal{ELH}	10,116
Cyc	\mathcal{LHF}	25,566
EClass-51En-1	\mathcal{LH}	76,977
Snomed-2	\mathcal{ELH}	182,869
Snomed-1	\mathcal{ELH}	223,260
Snomed	\mathcal{ELH}	379,691

- *Number of Threads*: To measure the scalability of our system, we have performed our experiments using different numbers of threads (1, 2, 4).
- *Partition Size*: The number of concepts (5, 25) that are assigned to every thread and are expected to be inserted by the corresponding thread. Similar to the number of threads, this parameter is also used to measure the scalability of our approach.
- *Number of Processors*: For the presented benchmarks we always had 8 processors or cores³ available.

Performance In order to test the effect of these parameters in our system, the benchmarks are run with different parameter values. The performance improvement is measured using the speedup factor which is defined as $Speedup_p = \frac{T_1}{T_p}$, where $Speedup_p$ is the speedup factor, and

- p is the number of threads. In the cluster environment we always had 8 cores available and never used more than 8 threads in our experiments, so, each thread can be considered as mapped to one core exclusively;
- T_1 is the CPU time for the sequential run using only one thread and one single partition containing all concept names to be inserted;
- T_p is the CPU time for the parallel run with p threads.

Effect of changing only the number of threads To measure the performance of the classifier in this case we selected EClass-51En-1 as our test case and ran the tests with a fixed partition size (5 or 25) but a different number of threads (2 and 4), as shown in Fig. 1. In the following we use $P_{threads,partition_size}$ to indicate a parallel multi-core setting where the subscripts give the number of cores available, the number of threads

³ For ease of presentation we use the terms *core* and *processor* as synonyms here.

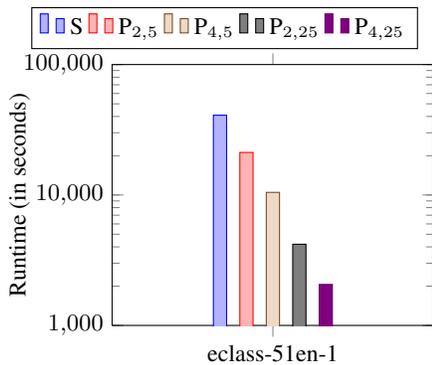


Fig. 1. Runtimes for eclass-51en-1 using 5 settings: S (sequential), P_{2,5}, P_{4,5}, P_{2,25}, P_{4,25} ($P_{threads,partition_size}$)

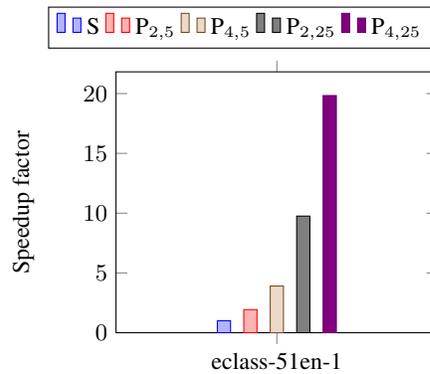


Fig. 2. Speedup for eclass-51en-1 from Fig. 1

created, and the partitions size used (from left to right). In the test cases P_{2,5} and P_{4,5}, we get an ideal speedup proportional to the number of threads, as shown in Fig. 2. As we can see, doubling the number of threads from S to P_{2,5} and to P_{4,5}, each time doubles the speedup, in other words, decreases the CPU time by the number of threads. This is the ideal speedup that we were expecting to happen.

Comparing the test cases S, P_{2,25}, and P_{4,25}, we get an even better speedup, also shown in Fig. 1 and 2. In this case, the CPU time decreases almost to $\frac{1}{10}$ compared to the sequential case (S). This speedup is due to a combination of the partition size as well as the cache effect and results from the different memory hierarchies of a cluster with modern computers. When we increase the number of threads to 4, the speedup is again proportional to the number of threads and this is what we expected. Here, by doubling the number of threads, the speedup doubles.

Effect of changing only partition sizes The performance of the classifier in this case for EClass-51En-1 is also shown in Fig. 1 with a fixed number of threads (2 or 4) but different partition sizes (5 or 25). When using 2 threads, compared to case S, we get the ideal speedup for P_{2,5}, as shown in Fig. 2. As we can see, doubling the number of threads, doubles the speedup, in other words, decreases the CPU time by half. This is the ideal case which is what we were expecting to happen. Again, compared to case S if the partition size is increased to 25, it shows the same speedup as shown in Fig. 2. In this case, the CPU time decreases almost to $\frac{1}{5}$ compared to the previous case.

In the scenario with 4 threads, we get a corresponding speedup, as shown in Fig. 2. In this case, the CPU time decreases to almost $\frac{1}{10}$ compared to the sequential case. This speedup again is due to a combination of the number of threads as well as the cache effect. When we increase the partition size to 25, the speedup is what we expected. Here, by multiplying the partition size by 5, the speedup is multiplied by five too.

Increasing the partition size, means that more concepts are assigned to one thread; therefore, all the related concepts are inserted into the taxonomy by one thread. Hence, increasing the partition size, reduces the number of corrections.

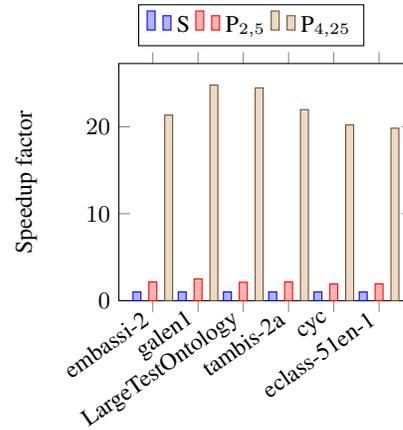
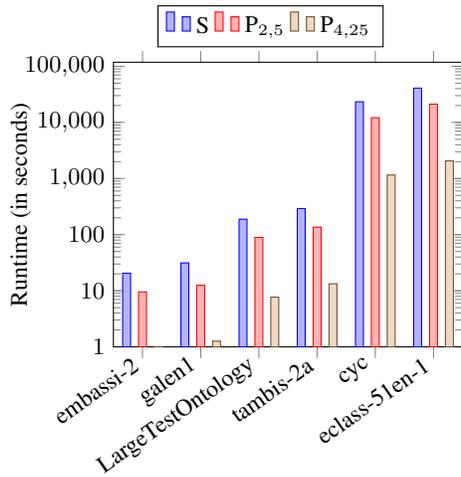


Fig. 3. Runtimes for ontologies using 3 settings: S (sequential), P_{2,5}, P_{4,25}

Fig. 4. Speedup for ontologies from Fig. 3

Effect of increasing both the number of threads and the partition size In this scenario, we measured the CPU time when increasing both the number of threads and the partition size. In Fig. 3 and 4, our test suite includes the ontologies Embassy-2, Galen1, LargeTestOntology, Tambis-2a, Cyc, and EClass-51En-1. The CPU time for each test case is shown in Fig. 3 and the speedup factor for each experiment is depicted in Fig. 4. As the results show, in the scenario with 2 threads and partition size 5, the speedup doubles compared to the sequential case and is around 2 and this is what we were expecting. When we increase the number of threads as well as the partition size, for the scenario with 4 threads and partition size 25, the CPU time decreases dramatically and therefore the speedup factor is above 20 for most test cases. This is more than a linear speedup, and it is the result of increasing the thread number as well as partition size together with the cache effect. The highest speedup factor is reported with test case Galen1.

Experiment on very large ontologies We selected 3 Snomed variants as very large ontologies with more than 150,000 concepts. Snomed-2 with 182,869 concepts, Snomed-1 with 223,260 concepts, and Snomed with 379,691 concepts were included in our tests. Fig. 7 shows an excellent improvement of CPU time for the parallel over the sequential case. In Fig. 8, the speedup factor is almost 2, which the expected behavior. The best speedup factor is observed for test case Snomed.

Observation on the increase of size of ontologies We chose Cyc, EClass-51en-1, Snomed-1, Snomed-2, and Snomed as test cases. Here, as shown in Fig. 5 and 7, in a parallel setting with 2 threads, the CPU time is divided by 2 compared to the sequential case. The speedup, shown in Fig. 6 and 8, is linear and is consistent for our benchmark ontologies even when the size of the ontologies increases.

Overall, the overhead is mostly determined by the quality of the told subsumers and disjoints information, the imposed order of traversal within a partitioning, the division of the ordered concept list into partitions, and the number of corrections which have been taken place (varied between 0.5% and 3% of the ontology size; depends on the

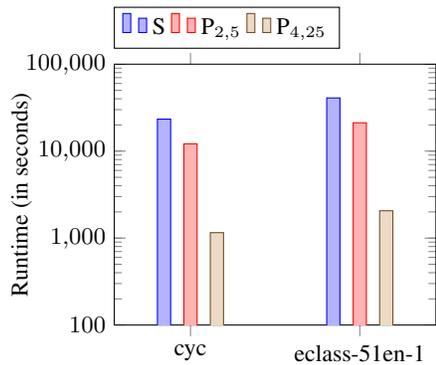


Fig. 5. Runtimes for cyc and eclass-51en-1 using 3 settings: S (sequential), P_{2,5}, P_{4,25}

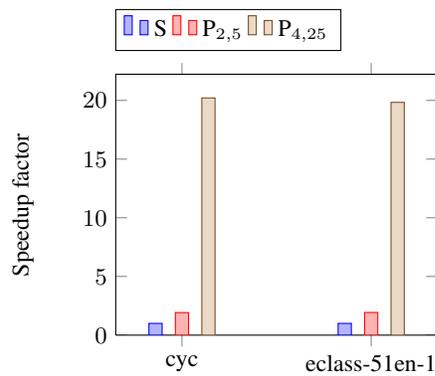


Fig. 6. Speedup for ontologies from Fig. 5

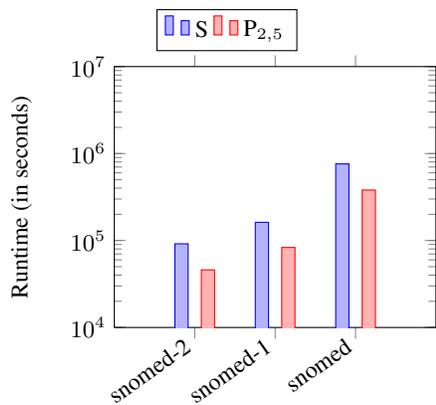


Fig. 7. Runtimes for snomed using 2 settings: S (sequential), P_{2,5}

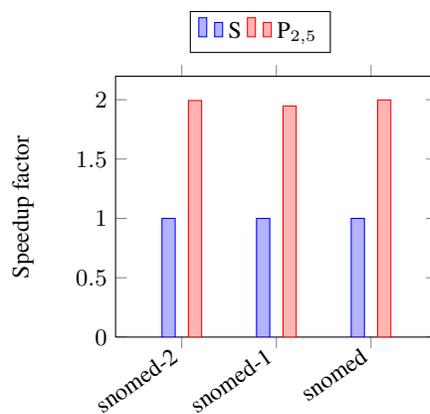


Fig. 8. Speedup for ontologies from Fig. 7

structure of ontology as well as the number of threads and partition size). In general, one should try to insert nodes as close as possible to their final order in the tree using a top to bottom strategy.

In Concurrent TBox Classifier no optimization techniques for classification have been implemented. For instance, there are well-known optimizations which can avoid subsumption tests or eliminate the bottom search for some DL languages or decrease the number of bottom searches in general. Of course, our system is not competitive at all compared to highly optimized DL reasoners or special-purpose reasoners designed to take advantage of the characteristics of the \mathcal{EL} fragment (e.g., see [7]). In our case, we can easily classify ontologies that are outside of the \mathcal{EL} fragment.

4 Conclusion

In this paper, we have shown an excellent scalable technique for concurrent OWL ontology classification. The explained architecture, which proposes lock-free algorithms with limited synchronization, utilizes concurrency in a multi-core environment. The experimental results show the effectiveness of our algorithms. We can say that this work appears to be the first which documents significant performance improvements in a multi-core environment using real-world benchmarks for ontologies of various DL complexities.

References

1. Aslani, M., Haarslev, V.: Parallel TBox classification in description logics - first experimental results. In: Proceedings of the 19th European Conference on Artificial Intelligence - ECAI 2010, Lisbon, Portugal, Aug. 16-20, 2010. pp. 485–490 (2010)
2. Baader, F., Franconi, E., Hollunder, B., Nebel, B., Profitlich, H.: An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence* 4(2), 109–132 (1994)
3. Bergmann, F., Quantz, J.: Parallelizing description logics. In: Proc. of 19th Ann. German Conf. on Artificial Intelligence. pp. 137–148. LNCS, Springer-Verlag (1995)
4. Haarslev, V., Möller, R.: RACER system description. In: Proc. of the Int. Joint Conf. on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy. pp. 701–705. LNCS (Jun 2001)
5. Haarslev, V., Möller, R., Turhan, A.Y.: Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In: Proc. of the Int. Joint Conf. on Automated Reasoning, IJCAR'2001, June 18-23, Siena, Italy. pp. 61–75 (2001)
6. Hogan, A., Pan, J., Polleres, A., Decker, S.: SAOR: template rule optimisations for distributed reasoning over 1 billion linked data triples. In: Proc. 9th Int. Semantic Web Conf. pp. 337–353 (2010)
7. Kazakov, Y., Krötzsch, M., Simancik, F.: Concurrent classification of EL ontologies. In: Proc. of the 10th Int. Semantic Web Conf. pp. 305–320 (2011)
8. Kotoulas, S., Oren, E., van Harmelen, F.: Mind the data skew: distributed inferencing by speeddating in elastic regions. In: Proc. 19th Int. Conf. on World Wide Web. pp. 531–540 (2010)
9. Liebig, T., Müller, F.: Parallelizing tableaux-based description logic reasoning. In: Proc. of 3rd Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS '07), Vilamoura, Portugal, Nov 27. LNCS, vol. 4806, pp. 1135–1144. Springer-Verlag (2007)
10. Mutharaju, R., Maier, F., Hitzler, P.: A MapReduce algorithm for EL+. In: Proc. 23rd Int. Workshop on Description Logics. pp. 464–474 (2010)
11. Schlicht, A., Stuckenschmidt, H.: Distributed resolution for expressive ontology networks. In: Web Reasoning and Rule Systems, 3rd Int. Conf. (RR 2009), Chantilly, VA, USA, Oct. 25-26, 2009. pp. 87–101 (2009)
12. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: WebPIE: a webscale parallel inference engine using mapreduce. In: *J. of Web Semantics* (2011)
13. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using MapReduce. In: International Semantic Web Conference. pp. 634–649 (2009)
14. Weaver, J., Hendler, J.: Parallel materialization of the finite RDFS closure for hundreds of millions of triples. In: Proc. 8th Int. Semantic Web Conf. pp. 87–101 (2009)

Non-Gödel Negation Makes Unwitnessed Consistency Undecidable*

Stefan Borgwardt and Rafael Peñaloza
{stefborg,penaloza}@tcs.inf.tu-dresden.de

Theoretical Computer Science, TU Dresden, Germany

Abstract. Recent results show that ontology consistency is undecidable for a wide variety of fuzzy Description Logics (DLs). Most notably, undecidability arises for a family of inexpressive fuzzy DLs using only conjunction, existential restrictions, and residual negation, even if the ontology itself is crisp. All those results depend on restricting reasoning to witnessed models. In this paper, we show that ontology consistency for inexpressive fuzzy DLs using any t-norm starting with the Łukasiewicz t-norm is also undecidable w.r.t. general models.

1 Introduction

Fuzzy Description Logics (DLs) extend the semantics of classical DLs by allowing a set of values, typically the real interval $[0, 1]$, to serve as truth degrees. They were introduced for representing and reasoning with vague or imprecise knowledge in a formal way [14,15]. While several decision procedures for fuzzy DLs exist, they usually rely on a restriction of the expressivity: either by allowing only finitely-valued semantics [5,7] or by limiting the terminological knowledge to be acyclic or unfoldable [11,9,4]. Indeed, the only algorithms for fuzzy \mathcal{ALC} with general ontologies are based on the very simple Gödel semantics [13,14,15].

It was recently shown that decision procedures for more expressive fuzzy DLs cannot exist since consistency of fuzzy \mathcal{ALC} ontologies becomes undecidable in the presence of GCIs whenever any continuous t-norm different from the Gödel t-norm is used [1,2,3,10,8]. Not all the constructors from \mathcal{ALC} are needed for proving undecidability. In particular, it was shown in [8] that for a family of t-norms—those t-norms “starting” with the Łukasiewicz t-norm—consistency is undecidable in the logic \mathfrak{NEL} , which allows only the constructors conjunction, existential restriction, and (residual) negation, even if all the axioms are crisp.

The crux of these undecidability results is that they all depend on restricting reasoning to the class of witnessed models. Since the existing reasoning algorithms are based also on witnessed models,¹ this semantics was a natural choice. However, it could be the case that witnessed models are the cause of undecidability and consistency w.r.t. general models is decidable.

* Partially supported by the DFG under grant BA 1122/17-1 and in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

¹ In fact, witnessed models were introduced in [11] to correct the methods from [15].

Although possible, such a scenario would be very surprising since for first-order fuzzy logic, reasoning w.r.t. general models is typically harder than reasoning w.r.t. witnessed models [12]. In this paper we show that this is also the case for the logic \mathfrak{NEL} over any t-norm starting with the Łukasiewicz t-norm: consistency of crisp ontologies w.r.t. general models is undecidable.

2 Preliminaries

We briefly introduce the family of t-norms starting with the Łukasiewicz t-norm, which will provide the semantics for the fuzzy DLs $\mathfrak{t}^{(0,b)}\text{-NEL}$.

2.1 The Łukasiewicz t-norm

Mathematical fuzzy logic generalizes the classical logical operators to deal with truth degrees from the interval $[0, 1]$. A *t-norm* is an associative and commutative binary operator on $[0, 1]$ that has 1 as its unit and is monotonic in both arguments. If a t-norm \otimes is continuous, then there is a unique binary operator \Rightarrow , called the *residuum*, that satisfies $z \leq x \Rightarrow y$ iff $x \otimes z \leq y$ for all $x, y, z \in [0, 1]$. In that case, for all $x, y \in [0, 1]$ we have (i) $x \Rightarrow y = 1$ iff $x \leq y$ and (ii) $1 \Rightarrow y = y$. Another useful operator is the unary *residual negation* $\ominus x := x \Rightarrow 0$.

We focus on the family of t-norms *starting with* the Łukasiewicz t-norm. The Łukasiewicz t-norm, defined by $x \otimes y = \max\{0, x + y - 1\}$, has the residuum $x \Rightarrow y = \min\{1, 1 - x + y\}$ and residual negation $\ominus x = 1 - x$. A t-norm \otimes *starts with* the Łukasiewicz t-norm if there is a $b \in (0, 1]$ such that for all $x, y \in [0, b]$ we have $x \otimes y = \max\{0, x + y - b\}$. Intuitively, such a t-norm behaves like a scaled-down version of the Łukasiewicz t-norm in the interval $[0, b]$.

All continuous t-norms that do not start with the Łukasiewicz t-norm have one characteristic in common: their residual negation is always the *Gödel negation* that simply maps 0 to 1 and every other truth degree to 0. In contrast, those that do start with the Łukasiewicz t-norm do not have the Gödel negation, but rather a scaled-down version of the Łukasiewicz negation, also known as the *involutive negation*. To be more precise, the residuum and residual negation of a t-norm starting with the Łukasiewicz t-norm satisfy $x \Rightarrow y = b - x + y$ whenever $0 \leq y < x \leq b$, and

$$\ominus x = \begin{cases} 1 & \text{if } x = 0 \\ b - x & \text{if } 0 < x \leq b \\ 0 & \text{otherwise.} \end{cases}$$

For the rest of this paper, we assume that \otimes is a t-norm starting with the Łukasiewicz t-norm in the interval $[0, b]$, for some arbitrary but fixed $b \in (0, 1]$, and \Rightarrow, \ominus are its residuum and residual negation, respectively.

2.2 The Fuzzy Description Logic $\mathfrak{t}^{(0,b)}\text{-NEL}$

Syntactically, $\mathfrak{t}^{(0,b)}\text{-NEL}$ extends the DL \mathcal{EL} with the unary *residual negation* constructor \boxminus . More precisely, concepts are built using the syntactic rule

$C ::= A \mid \top \mid C \sqcap D \mid \exists C \mid \exists r.C$. The semantics of $\mathfrak{L}^{(0,b)}\text{-}\mathfrak{N}\mathcal{E}\mathcal{L}$ is based on *interpretations* $\mathcal{I} = (\mathcal{D}^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a *domain* $\mathcal{D}^{\mathcal{I}}$ and an interpretation function that maps each concept name A to a function $A^{\mathcal{I}} : \mathcal{D}^{\mathcal{I}} \rightarrow [0, 1]$, each role name r to $r^{\mathcal{I}} : \mathcal{D}^{\mathcal{I}} \times \mathcal{D}^{\mathcal{I}} \rightarrow [0, 1]$, and each individual name a to an element $a^{\mathcal{I}}$ of $\mathcal{D}^{\mathcal{I}}$. This function is extended to complex concepts as follows: $\top^{\mathcal{I}}(x) = 1$, $(C \sqcap D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \otimes D^{\mathcal{I}}(x)$, $(\exists C)^{\mathcal{I}}(x) = \oplus C^{\mathcal{I}}(x)$, and $(\exists r.C)^{\mathcal{I}}(x) = \sup_{y \in \mathcal{D}^{\mathcal{I}}} r^{\mathcal{I}}(x, y) \otimes C^{\mathcal{I}}(y)$ for all $x \in \mathcal{D}^{\mathcal{I}}$.

In contrast to traditional semantics based on witnessed models, it is possible to have an interpretation \mathcal{I} such that $(\exists r.\top)^{\mathcal{I}}(x) = 1$ but where there is no $y \in \mathcal{D}^{\mathcal{I}}$ with $r^{\mathcal{I}}(x, y) = 1$. It can only be guaranteed that x has r -successors with degrees arbitrarily close to 1. This fact will produce some technical difficulties for our undecidability proof, since we it is not possible to guarantee a specific degree of membership for an r -successor of a given node (see Section 3.5).

We will use the following abbreviations: $C^0 := \top$, and $C^{k+1} := C^k \sqcap C$ for any $k \in \mathbb{N}$; $C \rightarrow D := \exists(C \sqcap \exists D)$; and $C \boxtimes D := (C \sqcap D) \rightarrow D$. It can easily be verified that, whenever $C^{\mathcal{I}}(x), D^{\mathcal{I}}(x) \in [0, b]$ for an interpretation \mathcal{I} and $x \in \mathcal{D}^{\mathcal{I}}$, then we have $(C^k)^{\mathcal{I}}(x) = \max\{0, k(C^{\mathcal{I}}(x) - b) + b\}$ for every $k > 0$, $(C \rightarrow D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)$, and $(C \boxtimes D)^{\mathcal{I}}(x) = \min\{C^{\mathcal{I}}(x), D^{\mathcal{I}}(x)\}$.

An *ontology* is a finite set of assertions of the form $a : C$ or $(a, b) : r$ and GCIs of the form $C \sqsubseteq D$, where a, b are individual names, and C, D are concepts. An interpretation \mathcal{I} *satisfies* the assertion $a : C$ (resp. $(a, b) : r$) if $C^{\mathcal{I}}(a^{\mathcal{I}}) = 1$ (resp. $r^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) = 1$); it *satisfies* the GCI $C \sqsubseteq D$ if $C^{\mathcal{I}}(x) \leq D^{\mathcal{I}}(x)$ for every $x \in \mathcal{D}^{\mathcal{I}}$. The expression $C \equiv D$ abbreviates the two GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$. It follows that an interpretation \mathcal{I} satisfies $C \equiv D$ iff $C^{\mathcal{I}}(x) = D^{\mathcal{I}}(x)$ for every $x \in \mathcal{D}^{\mathcal{I}}$. It is a *model* of an ontology \mathcal{O} if it satisfies all the axioms in \mathcal{O} . An ontology is called *consistent* if it has a model.

Notice that we consider only *crisp* axioms; i.e. they contain no associated fuzzy degree. We will show that reasoning with this restricted form of ontologies, which is the same used for classical (crisp) DLs, is already undecidable.

To show the undecidability of ontology consistency in $\mathfrak{L}^{(0,b)}\text{-}\mathfrak{N}\mathcal{E}\mathcal{L}$, we will use a reduction from the Post Correspondence Problem (PCP). We first show how to encode an instance of the PCP into an $\mathfrak{L}^{(0,b)}\text{-}\mathfrak{N}\mathcal{E}\mathcal{L}$ ontology, and then describe how to use ontology consistency to solve the decision problem.

3 Encoding the PCP

We reduce a variant of the undecidable PCP to the consistency problem in $\mathfrak{L}^{(0,b)}\text{-}\mathfrak{N}\mathcal{E}\mathcal{L}$, using an idea that has been applied before to show undecidability of fuzzy DLs. The goal is to construct an ontology $\mathcal{O}_{\mathcal{P}}$ whose models contain the complete search tree for a solution to a given instance \mathcal{P} of the PCP.

Definition 1 (PCP). *An instance \mathcal{P} of the Post Correspondence Problem is a finite set of pairs $\{(v_1, w_1), \dots, (v_n, w_n)\}$ of words over an alphabet Σ . A solution for \mathcal{P} is a sequence $i_1 \dots i_k \in \mathcal{N}^*$ such that $v_1 v_{i_1} \dots v_{i_k} = w_1 w_{i_1} \dots w_{i_k}$.*

We assume w.l.o.g. that the alphabet Σ is of the form $\{1, \dots, s\}$ for some natural number $s \geq 4$, and \mathcal{N} denotes the index set $\{1, \dots, n\}$. For $\nu = i_1 \cdots i_k \in \mathcal{N}^*$, we denote $v_1 v_{i_1} \cdots v_{i_k}$ by v_ν and $w_1 w_{i_1} \cdots w_{i_k}$ by w_ν .

The *search tree* \mathcal{T} of an instance \mathcal{P} of the PCP is a tree over the domain \mathcal{N}^* , in which each node $\nu \in \mathcal{N}^*$ is labeled by v_ν and w_ν . In order to represent this search tree in an $\mathfrak{L}^{(0,b)}$ - \mathfrak{NEL} -interpretation, we need an appropriate encoding of words from Σ^* into the interval $[0, 1]$. For technical reasons caused by the use of general semantics, we cannot guarantee that a specific real number will be used at a given node of the tree, but have to allow for a bounded error. Thus, a word can be encoded by any number from a given interval, as described next.

For $v = \alpha_1 \cdots \alpha_m$ with $\alpha_1, \dots, \alpha_m \in \Sigma$, \overleftarrow{v} denotes $\alpha_m \cdots \alpha_1$. The *encoding* $\text{enc}(v)$ of v is the number $0.\overleftarrow{v}$, in base $\beta := s + 2$. In particular, $\text{enc}(\varepsilon) := 0$. For $p \in [0, 1)$, the interval $\text{Err}_n(p)$ contains all numbers of the form $b(p + e) \in [0, b)$ with an *error term* e with $|e| < \beta^{-n}$. The set $\text{Enc}(v)$ of *bounded-error encodings* of a word $v \in \Sigma^+$ is defined as the intersection of $\text{Err}_{|v|+2}(\text{enc}(v))$ with $[0, b)$. The idea of these bounded-error encodings is to keep the error small enough to avoid overlapping. This way, we can decide whether two different real numbers are just different encoding of the same word; this is the case whenever they belong to the same error-bounded encoding.

Lemma 2. *Let $v, w \in \Sigma^+$, $p \in \text{Enc}(v)$, $q \in \text{Enc}(w)$, $g \in \text{Err}_{|v|+4}(\beta^{-(|v|+2)})$, and $h \in \text{Err}_{|w|+4}(\beta^{-(|w|+2)})$. Then $v = w$ iff $|p - q| < 3 \max\{g, h\}$.*

Proof. We can assume w.l.o.g. that $|v| = l \leq m = |w|$. It then follows that $3 \max\{g, h\} = 3b\beta^{-(l+2)} + e$ with $|e| < 3b\beta^{-(l+4)} < b\beta^{-(l+3)}$. Additionally, $p = \text{benc}(v) + e_1$ and $q = \text{benc}(w) + e_2$ with $|e_1| < b\beta^{-(l+2)}$ and $|e_2| < b\beta^{-(m+2)}$, and thus $|e_1 - e_2| < 2b\beta^{-(l+2)}$. If $v = w$, then

$$|p - q| = |e_1 - e_2| < 2b\beta^{-(l+2)} < 3b\beta^{-(l+2)} + e = 3 \max\{g, h\}.$$

If $v \neq w$, then $\text{enc}(v)$ and $\text{enc}(w)$ must differ at least in the $(l+1)$ -th digit. Thus, $|\text{enc}(v) - \text{enc}(w)| \geq \beta^{-(l+1)}$. This implies that

$$|p - q| \geq b|\text{enc}(v) - \text{enc}(w)| - |e_1 - e_2| > (\beta - 2)b\beta^{-(l+2)} \geq 4b\beta^{-(l+2)},$$

and hence $|p - q| > 3b\beta^{-(l+2)} + e = 3 \max\{g, h\}$. \square

We now construct an ontology $\mathcal{O}_{\mathcal{P}}$ whose models encode the search tree of \mathcal{P} . More precisely, it ensures that the concept names V and W will have values in $\text{Enc}(v_\nu)$ and $\text{Enc}(w_\nu)$, respectively, at every domain element that encodes the node $\nu \in \mathcal{N}^*$ of the search tree \mathcal{T} . The concept name G will have a value in $\text{Err}_{|v_\nu|+4}(1 - \beta^{-(|v_\nu|+2)})$, and similarly for H . These values provide a bound on the difference of the encoding of two words of a given length, and will be used to detect whether V and W encode the same word (see Lemma 2).

The search tree is encoded in the following way. First, we ensure that at every element of the domain the concepts V_i and W_i are interpreted as the encodings of the words v_i, w_i , respectively, without any error terms (Section 3.1).

They will be used to generate the encodings of the successor words $v_{\nu i} = v_{\nu}v_i$ from an encoding of v_{ν} , for every $\nu \in \mathcal{N}^*$ and $i \in \mathcal{N}$. We also use similar constants to update the values of G and H , which encode the error bounds introduced by the lack of witnessed models, from each node to its successor. Afterwards, we initialize the interpretation of all the relevant concepts at the root of the tree, identified by the individual name a_0 (Section 3.2). We have to ensure, e.g. that $V^{\mathcal{I}}(a_0^{\mathcal{I}})$ is an encoding of the word $v_{\varepsilon} = v_1$. We then describe the concatenation of encoded constant words to given encodings (Section 3.3), the creation of successors with large enough role degrees (Section 3.4), and the transfer of the concatenated encodings to these successors (Section 3.5).

3.1 Encoding Global Constants

We use some auxiliary values that will be constant along the whole search tree; these are helpful for producing the error-bounded encodings of the words v_{ν} and w_{ν} . First, we store the encoding of the words $v_1, \dots, v_n, w_1, \dots, w_n$ using the concept names $V_1, \dots, V_n, W_1, \dots, W_n$, respectively. More precisely, every model \mathcal{I} of $\mathcal{O}_{\mathcal{P}}$ should satisfy $V_i^{\mathcal{I}}(x) = \text{benc}(v_i)$ and $W_i^{\mathcal{I}}(x) = \text{benc}(w_i)$ for all $x \in \mathcal{D}^{\mathcal{I}}$ and $i \in \mathcal{N}$.

Let l be the length of the word v_i . We use the axioms

$$A_l^{\beta^l} \equiv \exists A_l^{\beta^l} \text{ and } \exists V_i \equiv A_l^{2\overleftarrow{v}_i},$$

where A_l is an auxiliary concept name. Let \mathcal{I} be a model of these axioms and $x \in \mathcal{D}^{\mathcal{I}}$. If $l = 0$, the second axiom implies that $V_i^{\mathcal{I}}(x) = \text{benc}(\varepsilon) = 0$. If $l > 0$, the first axiom enforces $\beta^l(A_l^{\mathcal{I}}(x) - b) + b = -\beta^l(A_l^{\mathcal{I}}(x) - b)$; thus $A_l^{\mathcal{I}}(x) = b(1 - \frac{1}{2\beta^l})$. Using the second axiom, we derive that $\exists V_i^{\mathcal{I}}(x) = \max\{0, b - \frac{2b\overleftarrow{v}_i}{2\beta^l}\}$. Since $\frac{\overleftarrow{v}_i}{\beta^l} = 0.\overleftarrow{v}_i < 1$, we have $V_i^{\mathcal{I}}(x) = \text{benc}(v_i)$.

We also encode the constant words g_i and h_i that yield the numbers $\beta^{|v_i|} - 1$ and $\beta^{|w_i|} - 1$, respectively, when read in base $\beta = s + 2$. That is, g_i is the concatenation of the symbol $(s + 1)$ with itself $|v_i|$ times, and analogously for h_i . We store them in the concept names G_i and H_i , respectively. They will be used to update the values of G and H in the search tree (see Section 3.5). The axioms $A_{|v_i|}^{\beta^{|v_i|}} \equiv \exists A_{|v_i|}^{\beta^{|v_i|}}$ and $\exists G_i \equiv A_{|v_i|}^{2g_i}$ force G_i to always be $b(1 - \beta^{-|v_i|})$; an analogous construction can be used for H_i . Using the same idea, we encode the words g_0 and h_0 represented by the numbers $\beta^{|v_1|+2} - 1$ and $\beta^{|w_1|+2} - 1$ in the concept names G_0 and H_0 , respectively.

3.2 Initializing the Root

At the root of the search tree of \mathcal{P} , designated by the individual name a_0 , the values of V and W should be $\text{benc}(v_1)$ and $\text{benc}(w_1)$, respectively. The two concept assertions $a_0 : (V \rightarrow V_1) \sqcap (V_1 \rightarrow V)$ and $a_0 : (W \rightarrow W_1) \sqcap (W_1 \rightarrow W)$ ensure this. The concept names G and H , storing the error bounds, are initialized to $b(1 - \beta^{-(|v_1|+2)})$ and $b(1 - \beta^{-(|w_1|+2)})$ using $a_0 : (G \rightarrow G_0) \sqcap (G_0 \rightarrow G)$ and $a_0 : (H \rightarrow H_0) \sqcap (H_0 \rightarrow H)$; G_0 and H_0 were defined in the previous section.

3.3 Concatenating Constants

We now describe how to express the concatenation of a given encoding with a constant word v_i . Assume that, for a given interpretation \mathcal{I} and $x \in \mathcal{D}^{\mathcal{I}}$, the value $V^{\mathcal{I}}(x)$ is an element of $\text{Enc}(v_\nu)$ for some $\nu \in \mathcal{N}^*$. The goal is to ensure that, for a given $i \in \mathcal{N}$, the value $V_i^{\mathcal{I}}(x)$ is an element of $\text{Enc}(v_\nu v_i) = \text{Enc}(v_\nu v_i)$. We introduce the axioms

$$(\boxplus V_i'')^{\beta^l} \equiv \boxplus V \text{ and } \boxplus V_i' \equiv (\boxplus V_i'') \sqcap (\boxplus V_i),$$

where $l = |v_i|$, V_i'' is a new concept name, and $V_i^{\mathcal{I}}(x) = \text{benc}(v_i)$ (see Section 3.1).

Let \mathcal{I} be an interpretation that satisfies the first axiom. If $v_\nu = \varepsilon$, then $V^{\mathcal{I}}(x) = 0$, and thus $V_i^{\mathcal{I}}(x) = 0 = \beta^{-l}V^{\mathcal{I}}(x)$. If $v_\nu \neq \varepsilon$, then $\ominus V^{\mathcal{I}}(x) \in (0, b)$, which implies that $\ominus V_i^{\mathcal{I}}(x) \in (0, b)$ and $b - V^{\mathcal{I}}(x) = \beta^l(-V_i^{\mathcal{I}}(x)) + b$, and thus $V_i^{\mathcal{I}}(x) = \beta^{-l}V^{\mathcal{I}}(x)$. In both cases, $V_i^{\mathcal{I}}(x)$ equals $V^{\mathcal{I}}(x) = b(0.\overleftarrow{v}_\nu + e)$, shifted l digits to the right. By assumption, the error term e satisfies $|e| < \beta^{-(|v_\nu|+2)}$.

Let now \mathcal{I} also satisfy the second axiom. If either v_ν or v_i is ε , then $V_i^{\mathcal{I}}(x)$ is $V_i^{\mathcal{I}}(x)$ or $V^{\mathcal{I}}(x)$, respectively. In both cases, this expresses the concatenation of v_ν and v_i . If both v_ν and v_i are non-empty words, then

$$\ominus V_i^{\mathcal{I}}(x) = b \max\{0, 1 - \beta^{-l}(0.\overleftarrow{v}_\nu + e) - (0.\overleftarrow{v}_i)\} = b \max\{0, 1 - (0.\overleftarrow{v}_{v_i} + e')\}$$

with $|e'| = \beta^{-l}|e| < \beta^{-(|v_\nu|+2)}$. Thus, $0.\overleftarrow{v}_{v_i} + e' < 1$ and $V_i^{\mathcal{I}}(x) \in \text{Enc}(v_\nu v_i)$.

The values of the concept names G and H should also be updated. These values also have an error term, which is two orders of magnitude smaller than the encoding of the words from the search tree. Let $G^{\mathcal{I}}(x) = b(1 - \beta^{-(|v_\nu|+2)} + e)$ with $|e| < \beta^{-(|v_\nu|+4)}$, which corresponds to the word $(s+1) \cdots (s+1)$ of length $|v_\nu|+2$. We have to concatenate this to the word $(s+1) \cdots (s+1)$ of length $l := |v_i|$, stored in G_i . As above, the axioms $(\boxplus G_i'')^{\beta^l} \equiv \boxplus G$ and $\boxplus G_i' \equiv (\boxplus G_i'') \sqcap (\boxplus G_i)$ ensure that

$$G_i^{\mathcal{I}}(x) = b - b(1 - \beta^{-l}(1 - \beta^{-(|v_\nu|+2)} + e) - (1 - \beta^{-l})) = b(1 - \beta^{-(|v_\nu|+2)} + e')$$

with $|e'| < \beta^{-(|v_\nu|+4)}$.

3.4 Creating Successor Nodes

The axioms $\top \sqsubseteq \exists r_i. \top$ for each $i \in \mathcal{N}$ enforce the existence of r_i -successors to each node of the search tree. This is the main difference to the proof of undecidability for $\mathbf{L}^{(0,b)}\text{-}\mathfrak{NEL}$ w.r.t. witnessed models in [8]. In fact, in any witnessed model \mathcal{I} of these axioms we have for each $x \in \mathcal{D}^{\mathcal{I}}$ a $y \in \mathcal{D}^{\mathcal{I}}$ such that $r_i^{\mathcal{I}}(x, y) = 1$. In the more general setting considered here, only the following holds.

Lemma 3. *Let \mathcal{I} satisfy $\top \sqsubseteq \exists r_i. \top$. Then for each $x \in \mathcal{D}^{\mathcal{I}}$ and every error bound $e > 0$ there is a $y_e \in \mathcal{D}^{\mathcal{I}}$ such that $r_i^{\mathcal{I}}(x, y_e) > 1 - e$.*

This is the main reason why we need to allow for bounded errors in the encodings of the words. If we could always guarantee the existence of r_i -successors with degree 1, then the axioms presented in the next section could be used to transfer all values exactly. However, since only Lemma 3 holds, the transfer might lead to an additional error.

3.5 Transferring Values

In Section 3.2, we have ensured that every interpretation contains an individual that encodes the root of the search tree. We are also able to compute the values needed in the successors using the constants from Section 3.1 and the concatenation from Section 3.3. Intuitively, these values should now be transferred to the successors created in the previous section; however, in general this transfer of values from a node of the search tree \mathcal{T} to a successor node will incur an error on the transferred value. Thus, we have to make sure that this error does not grow beyond the bounds of our encoding.

We consider first the case that $b = 1$. We need to transfer the value of V'_i at a node to the value of V at some r_i -successor of this node. We show that, if $V_i^{\mathcal{I}}(x) = b(\text{enc}(v_{\nu_i}) + e')$ is a bounded-error encoding of v_{ν_i} , then the axioms

$$\exists r_i.(\exists V) \sqsubseteq \exists V'_i \text{ and } \exists r_i.V \sqsubseteq V'_i$$

ensure that the value of V is also a bounded-error encoding of this word at all r_i -successors with large enough role degree since the incurred error stays below $e := b(\beta^{-(|v_{\nu_i}|+2)} - |e'|) > 0$.

Lemma 4. *Let \mathcal{I} satisfy $\exists r.(\exists D) \sqsubseteq \exists C$ and $\exists r.D \sqsubseteq C$, $b = 1$, and $0 < e < \frac{1}{2}$. Then for all $x, y \in \mathcal{D}^{\mathcal{I}}$ with $r^{\mathcal{I}}(x, y) > 1 - e$ we have $|C^{\mathcal{I}}(x) - D^{\mathcal{I}}(y)| < e$.*

Proof. Let x and y be as above. The axioms force \mathcal{I} to satisfy

$$\begin{aligned} r^{\mathcal{I}}(x, y) \otimes \exists D^{\mathcal{I}}(y) &\leq \sup_{y' \in \mathcal{D}^{\mathcal{I}}} r^{\mathcal{I}}(x, y') \otimes \exists D^{\mathcal{I}}(y') \leq \exists C^{\mathcal{I}}(x), \text{ and} \\ r^{\mathcal{I}}(x, y) \otimes D^{\mathcal{I}}(y) &\leq \sup_{y' \in \mathcal{D}^{\mathcal{I}}} r^{\mathcal{I}}(x, y') \otimes D^{\mathcal{I}}(y') \leq C^{\mathcal{I}}(x). \end{aligned}$$

This implies that $r^{\mathcal{I}}(x, y) - D^{\mathcal{I}}(y) \leq \max\{0, r^{\mathcal{I}}(x, y) - D^{\mathcal{I}}(y)\} \leq 1 - C^{\mathcal{I}}(x)$ and $r^{\mathcal{I}}(x, y) + D^{\mathcal{I}}(y) - 1 \leq \max\{0, r^{\mathcal{I}}(x, y) + D^{\mathcal{I}}(y) - 1\} \leq C^{\mathcal{I}}(x)$. From the first inequality it follows that $C^{\mathcal{I}}(x) - D^{\mathcal{I}}(y) \leq 1 - r^{\mathcal{I}}(x, y) < e$ and the second one yields $D^{\mathcal{I}}(y) - C^{\mathcal{I}}(x) \leq 1 - r^{\mathcal{I}}(x, y) < e$. This shows that the absolute difference between $C^{\mathcal{I}}(x)$ and $D^{\mathcal{I}}(y)$ is always smaller than e . \square

To transfer the value of G'_i along r_i to G , we use the axioms $\exists r_i.(\exists G) \sqsubseteq \exists G'_i$ and $\exists r_i.G \sqsubseteq G'_i$. The error bound $e := b(\beta^{-(|v_{\nu_i}|+4)} - |e'|)$ ensures that the additional error to $G_i^{\mathcal{I}}(x) = b(1 - \beta^{-(|v_{\nu_i}|+2)} + e')$ is small enough.

In order to simultaneously satisfy all the above-mentioned restrictions in the r_i -successors, we use only those $y \in \mathcal{D}^{\mathcal{I}}$ with $r_i^{\mathcal{I}}(x, y) > 1 - e$, where e is the minimum of the individual error bounds for V'_i , W'_i , G'_i , and H'_i . This is not a problem since Lemma 3 shows that such a successor y always exists.

In the case that $b < 1$, the transfer of values is considerably easier. In fact, no error bounds are necessary in this case since encodings of words can be transferred without additional errors through any r_i -successor with degree $> b$.

Lemma 5. *Let $b < 1$ and assume that \mathcal{I} satisfies $\exists r.(\exists D) \sqsubseteq \exists C$ and $\exists r.D \sqsubseteq C$. If $x, y \in \mathcal{D}^{\mathcal{I}}$ with $C^{\mathcal{I}}(x) \in [0, b)$ and $r^{\mathcal{I}}(x, y) > b$, then $C^{\mathcal{I}}(x) = D^{\mathcal{I}}(y)$.*

Proof. We have $r^{\mathcal{I}}(x, y) \otimes \ominus D^{\mathcal{I}}(y) \leq \ominus C^{\mathcal{I}}(x)$ and $r^{\mathcal{I}}(x, y) \otimes D^{\mathcal{I}}(y) \leq C^{\mathcal{I}}(x)$. If $C^{\mathcal{I}}(x) = 0$, then $r^{\mathcal{I}}(x, y) \otimes D^{\mathcal{I}}(y) = 0$, and thus $D^{\mathcal{I}}(y) = 0$. Consider now the case that $C^{\mathcal{I}}(x) \in (0, b)$. If $D^{\mathcal{I}}(y) \geq b$, then $C^{\mathcal{I}}(x) \geq r^{\mathcal{I}}(x, y) \otimes D^{\mathcal{I}}(y) \geq b$, contradicting the assumption. Likewise, $D^{\mathcal{I}}(y) = 0$ implies $b - C^{\mathcal{I}}(x) \geq r^{\mathcal{I}}(x, y) > b$, which is also impossible. Thus, we must have $D^{\mathcal{I}}(y) \in (0, b)$, which implies $b - D^{\mathcal{I}}(y) \leq b - C^{\mathcal{I}}(x)$ and $D^{\mathcal{I}}(y) \leq C^{\mathcal{I}}(x)$, and hence $C^{\mathcal{I}}(x) = D^{\mathcal{I}}(y)$. \square

As before, Lemma 3 ensures that a successor with degree strictly greater than b always exists, which shows that we can always transfer all desired values exactly.

3.6 Canonical Model

From the constructions of Sections 3.1 to 3.5 we obtain the ontology $\mathcal{O}_{\mathcal{P}}$:

$$\begin{aligned} \mathcal{O}_{\mathcal{P}} &:= \mathcal{O}_0 \cup \mathcal{O}_{G_0:=g_0} \cup \mathcal{O}_{H_0:=h_0} \\ &\quad \cup \bigcup_{i=1}^n \mathcal{O}_{V_i:=v_i} \cup \mathcal{O}_{W_i:=w_i} \cup \mathcal{O}_{G_i:=g_i} \cup \mathcal{O}_{H_i:=h_i} \cup \mathcal{O}_{r_i} \\ &\quad \quad \cup \mathcal{O}_{V'_i=V \circ v_i} \cup \mathcal{O}_{W'_i=W \circ w_i} \cup \mathcal{O}_{V'_i \overset{r_i}{\rightsquigarrow} V} \cup \mathcal{O}_{W'_i \overset{r_i}{\rightsquigarrow} W} \\ &\quad \quad \cup \mathcal{O}_{G'_i=G \circ g_i} \cup \mathcal{O}_{H'_i=H \circ h_i} \cup \mathcal{O}_{G'_i \overset{r_i}{\rightsquigarrow} G} \cup \mathcal{O}_{H'_i \overset{r_i}{\rightsquigarrow} H}, \quad \text{where} \\ \mathcal{O}_0 &:= \{a_0 : (V \rightarrow V_1) \sqcap (V_1 \rightarrow V), a_0 : (W \rightarrow W_1) \sqcap (W_1 \rightarrow W)\} \cup \\ &\quad \{a_0 : (G \rightarrow G_0) \sqcap (G_0 \rightarrow G), a_0 : (H \rightarrow H_0) \sqcap (H_0 \rightarrow H)\}, \\ \mathcal{O}_{C:=u} &:= \{C_{|u|}^{\beta^{|u|}} \equiv \boxplus C_{|u|}^{\beta^{|u|}}, \boxplus C \equiv C_{|u|}^{2\overline{u}}\}, \\ \mathcal{O}_{D'=C \circ u} &:= \{(\boxplus D'')^{\beta^{|u|}} \equiv \boxplus C, \boxplus D' \equiv (\boxplus D'') \sqcap (\boxplus D)\}, \\ \mathcal{O}_r &:= \{\top \sqsubseteq \exists r. \top\}, \\ \mathcal{O}_{C \overset{r}{\rightsquigarrow} D} &:= \{\exists r. (\boxplus D) \sqsubseteq \boxplus C, \exists r. D \sqsubseteq C\}. \end{aligned}$$

A model of this ontology is given by the interpretation $\mathcal{I}_{\mathcal{P}} = (\mathcal{N}^*, \cdot^{\mathcal{I}_{\mathcal{P}}})$, where $\cdot^{\mathcal{I}_{\mathcal{P}}}$ is defined as follows:

- $a_0^{\mathcal{I}_{\mathcal{P}}} = \varepsilon$,
- $V^{\mathcal{I}_{\mathcal{P}}}(\nu) = \text{benc}(v_\nu)$, $V_i^{\mathcal{I}_{\mathcal{P}}}(\nu) = \text{benc}(v_i)$, $V'_i{}^{\mathcal{I}_{\mathcal{P}}}(\nu) = \text{benc}(v_{\nu i})$,
- $W^{\mathcal{I}_{\mathcal{P}}}(\nu) = \text{benc}(w_\nu)$, $W_i^{\mathcal{I}_{\mathcal{P}}}(\nu) = \text{benc}(w_i)$, $W'_i{}^{\mathcal{I}_{\mathcal{P}}}(\nu) = \text{benc}(w_{\nu i})$,
- $G^{\mathcal{I}_{\mathcal{P}}}(\nu) = b(1 - \beta^{-(|v_\nu|+2)})$, $H^{\mathcal{I}_{\mathcal{P}}}(\nu) = b(1 - \beta^{-(|w_\nu|+2)})$,
- $G_i^{\mathcal{I}_{\mathcal{P}}}(\nu) = b(1 - \beta^{-|v_i|})$, $H_i^{\mathcal{I}_{\mathcal{P}}}(\nu) = b(1 - \beta^{-|w_i|})$,
- $r_i^{\mathcal{I}_{\mathcal{P}}}(\nu, \nu i) = 1$ and $r_i^{\mathcal{I}_{\mathcal{P}}}(\nu, \nu') = 0$ if $\nu' \neq \nu i$.

Notice that the values of all the auxiliary concept names used in the construction are fully determined by these concept names. Intuitively, $\mathcal{I}_{\mathcal{P}}$ is an encoding of the search tree of \mathcal{P} . We now show that every model \mathcal{I} of $\mathcal{O}_{\mathcal{P}}$ encodes the search tree \mathcal{T} of \mathcal{P} under the following notion of encoding.

Definition 6. *Let \mathcal{I} be an interpretation, $x \in \mathcal{D}^{\mathcal{I}}$, and $\nu \in \mathcal{N}^*$. We say that \mathcal{I} at x encodes \mathcal{T} at ν if, for every $i \in \mathcal{N}$,*

- a) $V^{\mathcal{I}}(x) \in \text{Enc}(v_\nu)$, $W^{\mathcal{I}}(x) \in \text{Enc}(w_\nu)$,
b) $G^{\mathcal{I}}(x) \in \text{Err}_{|v_\nu|+4}(1 - \beta^{-(|v_\nu|+2)})$, $H^{\mathcal{I}}(x) \in \text{Err}_{|w_\nu|+4}(1 - \beta^{-(|w_\nu|+2)})$.

In the following, whenever we talk about these properties, we will only consider V and G since W and H can be treated in an analogous manner.

Lemma 7. *For every model \mathcal{I} of $\mathcal{O}_{\mathcal{P}}$ there is a function $f : \mathcal{N}^* \rightarrow \mathcal{D}^{\mathcal{I}}$ such that for all $\nu \in \mathcal{N}^*$, \mathcal{I} at $f(\nu)$ encodes \mathcal{T} at ν .*

Proof. We construct the function f by induction on $\nu \in \mathcal{N}^*$. For $\nu = \varepsilon$, observe that the values of V and G at $a_0^{\mathcal{I}}$ are forced by \mathcal{O}_0 to be $V_1^{\mathcal{I}}(\varepsilon) = \text{benc}(v_\varepsilon)$ and $G_0^{\mathcal{I}}(\varepsilon) = b(1 - \beta^{-(|v_\varepsilon|+2)})$, respectively. Thus, for $f(\nu) := a_0^{\mathcal{I}}$ the conditions are satisfied even without any error terms.

Assume now that \mathcal{I} at $f(\nu)$ encodes \mathcal{T} at ν and let $i \in \mathcal{N}$. As described in Sections 3.4 and 3.5, one can find $0 < e < \frac{1}{2}$ such that the values of V'_i and G'_i are transferred to V and G , respectively, without increasing the error bounds on the encoded values. Specifically, we can choose

$$e := \min_{i=1, \dots, n} \left\{ \frac{1}{3}, b(\beta^{-(|v_{\nu i}|+2)} - |e_{V'_i}|), b(\beta^{-(|v_{\nu i}|+4)} - |e_{G'_i}|), \dots \right\}$$

if $e_{V'_i}$ and $e_{G'_i}$ are the error terms of V'_i and G'_i , respectively. Lemmata 4 and 5 show that $V^{\mathcal{I}}(y)$ is a bounded-error encoding of $v_{\nu i}$ whenever $r_i^{\mathcal{I}}(x, y) > 1 - e$, and similarly for $G^{\mathcal{I}}(y)$. By Lemma 3, there is a $y_i \in \mathcal{D}^{\mathcal{I}}$ such that \mathcal{I} at y_i encodes \mathcal{T} at νi . We can thus define $f(\nu i) := y_i$ to satisfy the condition. \square

Thus, in any model of $\mathcal{O}_{\mathcal{P}}$ we can find an encoding of the search tree \mathcal{T} of \mathcal{P} . The canonical model $\mathcal{I}_{\mathcal{P}}$ is the special case in which all error terms are 0. In the next section we use this encoding to solve the instance \mathcal{P} of the PCP.

4 Finding a Solution

To decide whether \mathcal{P} has a solution, we use the additional ontology \mathcal{O}_{\neq} , consisting of the single axiom $(V \rightarrow W) \sqcap (W \rightarrow V) \sqsubseteq (G \boxtimes H)^3$. Recall that $G \boxtimes H$ is interpreted as the minimum of the values of G and H in the interval $[0, b]$.

Lemma 8. *\mathcal{P} has a solution iff $\mathcal{O}_{\mathcal{P}} \cup \mathcal{O}_{\neq}$ is inconsistent.*

Proof. If this ontology has no model, then in particular the canonical model $\mathcal{I}_{\mathcal{P}}$ of $\mathcal{O}_{\mathcal{P}}$ cannot satisfy \mathcal{O}_{\neq} . We thus have

$$(V \rightarrow W)^{\mathcal{I}_{\mathcal{P}}}(\nu) \otimes (W \rightarrow V)^{\mathcal{I}_{\mathcal{P}}}(\nu) > ((G \boxtimes H)^3)^{\mathcal{I}_{\mathcal{P}}}(\nu)$$

for some $\nu \in \mathcal{N}^*$. If $V^{\mathcal{I}_{\mathcal{P}}}(\nu) = W^{\mathcal{I}_{\mathcal{P}}}(\nu)$, then $v_\nu = w_\nu$, i.e. \mathcal{P} has a solution. We now assume without loss of generality that $V^{\mathcal{I}_{\mathcal{P}}}(\nu) < W^{\mathcal{I}_{\mathcal{P}}}(\nu)$, and thus

$$b - W^{\mathcal{I}_{\mathcal{P}}}(\nu) + V^{\mathcal{I}_{\mathcal{P}}}(\nu) > ((G \boxtimes H)^3)^{\mathcal{I}_{\mathcal{P}}}(\nu) = b - 3b \max\{\beta^{-(|v_\nu|+2)}, \beta^{-(|w_\nu|+2)}\}.$$

This implies that

$$|\text{enc}(w_\nu) - \text{enc}(v_\nu)| < 3b \max\{\beta^{-(|v_\nu|+2)}, \beta^{-(|w_\nu|+2)}\},$$

and by Lemma 2 we again have $v_\nu = w_\nu$.

Assume now that $\mathcal{O}_{\mathcal{P}} \cup \mathcal{O}_{\neq}$ has a model \mathcal{I} and let f be the function constructed in Lemma 7 and $\nu \in \mathcal{N}^*$. In particular, we have

$$((G \boxtimes H)^3)^{\mathcal{I}}(f(\nu)) = b - 3 \max\{g, h\} \in (0, b)$$

with $g \in \text{Err}_{|v_\nu|+4}(\beta^{-(|v_\nu|+2)})$ and $h \in \text{Err}_{|w_\nu|+4}(\beta^{-(|w_\nu|+2)})$. Since \mathcal{I} satisfies \mathcal{O}_{\neq} , it holds that

$$((V \rightarrow W) \sqcap (W \rightarrow V))^{\mathcal{I}}(f(\nu)) \leq ((G \boxtimes H)^3)^{\mathcal{I}}(f(\nu)).$$

If we assume w.l.o.g. that $V^{\mathcal{I}}(f(\nu)) \leq W^{\mathcal{I}}(f(\nu))$, then this implies

$$W^{\mathcal{I}}(f(\nu)) - V^{\mathcal{I}}(f(\nu)) \geq 3 \max\{g, h\},$$

and by Lemma 2, $v_\nu \neq w_\nu$. Since this holds for all $\nu \in \mathcal{N}^*$, \mathcal{P} has no solution. \square

A direct consequence of this lemma is the undecidability of ontology consistency in the fuzzy DL $\mathfrak{L}^{(0,b)}\text{-}\mathfrak{N}\mathcal{E}\mathcal{L}$.

Theorem 9. *Consistency of $\mathfrak{L}^{(0,b)}\text{-}\mathfrak{N}\mathcal{E}\mathcal{L}$ -ontologies is undecidable.*

In particular, this shows that ontology consistency in $\mathfrak{L}\text{-}\mathcal{A}\mathcal{L}\mathcal{C}$ is undecidable w.r.t. general models, even if the ontologies contain only crisp axioms.

5 Conclusions

We have shown that ontology consistency w.r.t. general models is undecidable in fuzzy DLs based on t-norms starting with the Łukasiewicz t-norm using only the constructors from $\mathcal{E}\mathcal{L}$ and residual negation. This was achieved by a modification of the undecidability proofs for these logics w.r.t. witnessed model semantics [8]. The main problem introduced by general semantics is that it is impossible to ensure that values are transferred exactly from a node to its role successors. To simulate the search tree for an instance of the PCP, we allow for a bounded error in the represented value. The bounds are small enough to ensure that different words can still be distinguished.

Ontology consistency is a central decision problem for DLs since other reasoning tasks like concept satisfiability or subsumption can be reduced to it. However, the converse reduction does not hold. It is thus natural to ask whether these other problems are also undecidable. Since our construction uses only crisp assertions that involve only one individual name, it follows that also concept satisfiability in $\mathfrak{L}^{(0,b)}\text{-}\mathfrak{N}\mathcal{E}\mathcal{L}$ is undecidable w.r.t. general models.

Together with the results from [6], this fully characterizes the decidability of consistency and satisfiability w.r.t. general models in all fuzzy DLs between

\otimes - \mathfrak{NEL} and \otimes - $\mathcal{SHOI}^{-\forall}$:² it is decidable (in EXPTIME) iff the t-norm \otimes does not start with the Łukasiewicz t-norm, i.e. the fuzzy DL has Gödel negation.

In future work we plan to extend the study of fuzzy DLs with general semantics, aiming towards a full characterization of their complexity properties, as has been done for witnessed models. In particular, we think that the presented approach using error bounds can be applied to modify other undecidability results for fuzzy DLs w.r.t. witnessed models, e.g. for Π - \mathcal{ALC} [8]. We also plan to generalize the framework presented in [8] to deal with general models.

References

1. Baader, F., Peñaloza, R.: Are fuzzy description logics with general concept inclusion axioms decidable? In: Proc. FUZZ-IEEE'11. pp. 1735–1742. IEEE Press (2011)
2. Baader, F., Peñaloza, R.: GCIs make reasoning in fuzzy DL with the product t-norm undecidable. In: Proc. DL'11. CEUR-WS, vol. 745, pp. 37–47 (2011)
3. Baader, F., Peñaloza, R.: On the undecidability of fuzzy description logics with GCIs and product t-norm. In: Proc. FroCoS'11, LNCS, vol. 6989, pp. 55–70. Springer (2011)
4. Bobillo, F., Straccia, U.: Fuzzy description logics with general t-norms and datatypes. Fuzzy Set. Syst. 160(23), 3382–3402 (2009)
5. Bobillo, F., Straccia, U.: Reasoning with the finitely many-valued Łukasiewicz fuzzy description logic \mathcal{SROIQ} . Inform. Sciences 181, 758–778 (2011)
6. Borgwardt, S., Distel, F., Peñaloza, R.: Gödel negation makes unwitnessed consistency crisp. In: Proc. DL'12. CEUR-WS (2012), to appear.
7. Borgwardt, S., Peñaloza, R.: Description logics over lattices with multi-valued ontologies. In: Walsh, T. (ed.) Proc. IJCAI'11. pp. 768–773. AAAI Press (2011)
8. Borgwardt, S., Peñaloza, R.: Undecidability of fuzzy description logics. In: Proc. KR'12. AAAI Press (2012), to appear.
9. Cerami, M., Esteva, F., Bou, F.: Decidability of a description logic over infinite-valued product logic. In: Proc. KR'10. pp. 203–213. AAAI Press (2010)
10. Cerami, M., Straccia, U.: On the undecidability of fuzzy description logics with GCIs with Łukasiewicz t-norm. Tech. rep., Computing Research Repository (2011), [arXiv:1107.4212v3](https://arxiv.org/abs/1107.4212v3) [cs.LG]. An extended version of this paper has been submitted to a journal.
11. Hájek, P.: Making fuzzy description logic more general. FSS 154(1), 1–15 (2005)
12. Hájek, P.: Arithmetical complexity of fuzzy predicate logics—a survey II. Ann. Pure Appl. Logic 161(2), 212–219 (2009)
13. Stoilos, G., Stamou, G.B., Pan, J.Z., Tzouvaras, V., Horrocks, I.: Reasoning with very expressive fuzzy description logics. JAIR 30, 273–320 (2007)
14. Straccia, U.: Reasoning within fuzzy description logics. JAIR 14, 137–166 (2001)
15. Tresp, C.B., Molitor, R.: A description logic for vague knowledge. In: Proc. ECAI'98. pp. 361–365. John Wiley and Sons (1998)

² \otimes - $\mathcal{SHOI}^{-\forall}$ extends \otimes - \mathfrak{NEL} by the constructors \sqcup , \rightarrow , inverse and transitive roles, fuzzy role hierarchies, and nominals.

ORM2 Encoding into Description Logic (Extended Abstract)

Enrico Franconi¹, Alessandro Mosca¹, and Dmitry Solomakhin¹

Free University of Bozen-Bolzano, Italy

¹*lastname@inf.unibz.it*

Abstract. The *Object Role Modelling* (ORM2) is a conceptual modelling approach combining both textual specifications and graphical language, similar to UML and ER, and adopted by Visual Studio, the integrated development environment designed by Microsoft. This paper introduces a new linear syntax and corresponding complete set-theoretic semantics for a generalization of ORM2 language. A core fragment of ORM2 is defined, for which a provably correct encoding into \mathcal{ALCQI} description logic is presented. Based on these results, an extensive and systematic critique of alternative approaches to the formalisation of ORM2 in (description) logics published so far is provided. A first prototype has been implemented, which offers a back-end for the automated support of consistency and entailment checks for ORM2 conceptual schemas along with its translation into \mathcal{ALCQI} knowledge bases.

1 Introduction

Automated support to enterprise modelling has increasingly become a subject of interest for organisations seeking solutions for storage, distribution and analysis of knowledge about business processes [6], and the main expectation from automated solutions built upon these approaches is the ability to automatically determine consistency of a business model. Despite existence of reasoning tools for Unified Modelling Language (UML) [2], its known weakness with regard to verbalisation of facts and constraints restricts its usage by domain experts [10]. Recently becoming popular ORM2 (‘Object Role Modelling 2’) is a graphical fact-oriented approach for modelling, transforming, and querying business domain information, which allows for a verbalisation in language readily understandable by non-technical users. Being domain expert oriented, the semantics of ORM2 differs from that of UML (e.g. permitting optionality of cardinality constraints) and thus makes ORM2 richer in its capacity to express business constraints [10].

The NIAM language (‘Natural-language Information Analysis Method’), ancestor ORM, has been equipped with an FOL-based semantics for the first time in 1989 [9]. Since then, despite the remarkable evolution in terms of expressivity and graphical notation that ORM2 has experienced, much less attention has been paid in the consequent development of appropriate formal foundations for the modelling language.

This paper addresses the main problem of providing a logic formalism, equipped with sound and complete reasoning services, that captures the expressiveness of ORM2. The first contribution of the paper is thus the introduction of a completely new linear

syntax and a set-theoretic semantics for ORM2 matching the usage patterns in the community. The new syntax can be used to express the full set of ORM2 graphical symbols introduced in [10]. The second contribution of the paper is driven by a practical objective. On the basis of well known results developed in the Description Logics (DLs) community, we identified a ‘core’ fragment of ORM2 that can be translated in a sound and complete way into the EXP-TIME-complete logic \mathcal{ALCQI} [1], through *n*-ary relations *reification*. On the basis of the results presented in the paper, a first prototype, built on top of available DL reasoners, has been implemented, which provides an automated support for schema consistency, entity/relations consistency check, and entailment verification for user-defined ORM2 statements.

The rest of the paper is organised as follows: Section 2 is about the introduction, through examples, of the ORM2 graphical notation and intended semantics in the framework of the fact modelling approach; Section 3 introduces the new linear syntax by means of expressing the example from the previous section in this syntax. The encoding of the corresponding set-theoretic semantics into the DL logic \mathcal{ALCQI} is the main topic of Section 4. Finally, Section 5 gives an overview of the implemented reasoning support prototype, its interface and gives an example of its usage.

2 Fact-oriented Modelling in ORM2

Basic ORM2 objects are: **entities** (e.g. a house or a car) and **values** (e.g. *character string* or *number*). Moreover, entities and values are described in terms of the **types** they belong to: a type (e.g. House, Car) is a set of possible instances. In order to avoid ambiguity among the possible instances of a given type, entities are identified also by means of a particular *reference mode* and a value. The roles played by the entities in a given domain are introduced by means of logical **predicates**; each predicate (or relation) has a given set of roles according to its arity. Each role is connected to exactly one object type, indicating that the role is played only by possible instances of that type.

The first step of the ORM2 design procedure thus concerns the specification of the relevant **object types** (i.e. entity and value types), predicates and reference modes. All the subsequent steps in the procedure mostly deal with the specification of static *constraints*. Let us consider the example in Fig. 1:

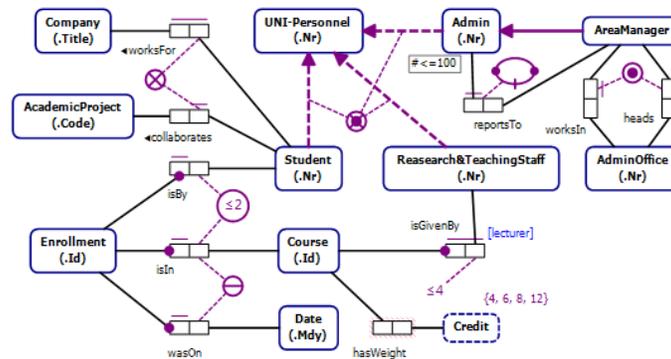


Fig. 1. A conceptual schema including an instantiation of most of the ORM2 constraints

The schema includes:

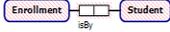
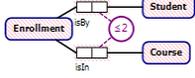
1. **Entity types:** Enrollment, Student, Date, ... ;
2. **Binary predicates:** isBy, wasOn, worksFor, ... ;
3. A user-defined **role name** [lecturer], for the role played by Research&TeachingStaff;
4. **Reference modes** for each entity: .ld, .Nr, .Mdy, ... ;
5. **Subtyping** links (depicted as thick arrows) indicating 'isa' relationships among types, and a constraint combination, called **partition**, made of an **exclusive** constraint (a circled 'X' for *mutual disjointness*), and a **total** constraint (a circled dot for complete coverage of the common super-type).
6. **Internal frequency occurrence** constraint indicating that *if* an instance of Research&TeachingStaff plays the role of being lecturer in the relation isGivenBy, it plays the role at most 4 times.
7. An **external frequency occurrence** that applies to roles played by Student and Course, meaning that 'Students are allowed to enroll in the same course *at most twice*'.
8. An **external uniqueness** constraint between the role played by Course in isIn and the role played by Date in wasOn, saying that 'For each combination of Course and Date, *at most one* Enrollment isIn that Course and wasOn that Date'.
9. A disjunctive mandatory 'circled dot', called **inclusive-or**, linking the roles played by AreaManager indicating that 'Each area manager *either* works in *or* heads (or *both*)'.
10. An **object cardinality** constraint forcing the number of the Admin instances to be less or equal to 100.
11. An **object type value** constraint indicating which values are allowed in Credit.
12. An **exclusion** constraint (depicted as circled 'X') between the roles played by Student in the relations worksFor and collaborates, expressing the fact that no student can play *both* these roles.
13. A **ring** constraint expressing that the relation reportsTo is *asymmetric*.

3 Proposed Formalisation of ORM2

As mentioned before, the modelling activity in ORM2 is supported by several tools that provide user friendly graphical interfaces to build complex conceptual schemas. However, none of the available design tools offers automated reasoning support on specific combinations of ORM2 constraints. The automated verification of *schema consistency* and *consistency of an object type* over a conceptual schema strictly depends on the possibility to perform reasoning and make inferences on it by means of a semantic-based logic representation of the schema itself.

With this goal in mind, we propose a linear syntax that fully covers the set of graphical symbols of ORM2. Table 1, using the example from previous section, shows how a new introduced syntax can be used to encode conceptual schemas that have been originally specified in graphical terms. For each construct ϕ in the syntax, its corresponding set-theoretic semantics expressed in relational algebra is also introduced in table 2 (where O denotes an object type).

Table 1. Constraints C1-C7 below represent a fragment of the schema from Fig. 1.

		ENTITYTYPES: {Enrollement, Student, ...} VALUETYPES: {Credit, Student-Nr, ...} RELATIONS: {isIn, isBy, collaborates, student-Nr, ...}
C1.		TYPE(isBy.enrollment, Enrollement) TYPE(isBy.student, Student)
C2.		MAND((isBy.enrollment), Enrollement)
C3.		FREQ((isBy.student, isIn.course), (isBy.enrollment, isIn.enrollment), (1, 2))
C4.		O-SET _{Tot} ((R&TStaff, Student, Admin), UNI-Personnel) O-SET _{Ex} ((R&TStaff, Student, Admin), UNI-Personnel)
C5.		RING _{Asym} (reportsTo.sub, reportsTo.obj)
C6.		V-VAL(Credit)={4, 6, 8, 12}
C7.		O-CARD(Admin)=(0, 100)

The signature \mathcal{S} of the linear ORM2 syntax is made of:

- Disjoint sets \mathcal{E} and \mathcal{V} of *entity type* and *value type* symbols, respectively;
- a set \mathcal{R} of *relation* symbols and a set \mathcal{A} of corresponding *role* symbols;
- a set \mathcal{D} of *domain* symbols, and a set Λ of pairwise disjoint sets of values;
- for each $D \in \mathcal{D}$, an injective extension function $\Lambda_{(D)} : \mathcal{D} \rightarrow \Lambda$ associating each domain symbol D to an extension Λ_D ;
- a binary relation $\varrho \subseteq \mathcal{R} \times \mathcal{A}$ linking role symbols to relation symbols. We take the pair $R.a$ as the atomic elements of the syntax, and we call it *localised role*. Given a relation symbol R , $\varrho_R = \{R.a | R.a \in \varrho\}$ is the set of localised roles with respect to R ;
- for each relation symbol R , a bijection $\tau_R : \varrho_R \rightarrow [1..|\varrho_R|]$ mapping each element in ϱ_R to an element in the finite sequence of natural numbers $[1..|\varrho_R|]$. The mapping τ_R guarantees a correspondence between role components and argument positions in a relation.

Given the signature \mathcal{S} , an **ORM2 conceptual schema** Σ over \mathcal{S} includes a finite combination of the constructs in table 2.

4 Encoding in \mathcal{ALCQI}

With the main aim of relying on available reasoning tools to reason in an effective way on ORM2 schemas, we present here the encoding in the logic \mathcal{ALCQI} , for which tableaux-based reasoning algorithms with a tractable computational complexity have been developed [8,12]. The \mathcal{ALCQI} encoding has been devised through an intermediate translation step in the logic \mathcal{DLR} , where arbitrary n -ary relations are allowed [3].

Table 2. Linear Syntax and Semantics table

Syntax	Semantics
$\text{TYPE} \subseteq \varrho \times (\mathcal{E} \cup \mathcal{V})$	If $\text{TYPE}(R.a, O) \in \Sigma$ then $\Pi_{R.a} R^I \subseteq O^I$
$\text{FREQ} \subseteq \varphi(\varrho) \times \varphi(\varrho) \times (\mathbb{N} \times (\mathbb{N} \cup \{\infty\}))$	If $\text{FREQ}(\{R^1.a_1, \dots, R^1.a_n, \dots, R^k.a_1, \dots, R^k.a_m\}, \{R^1.a, \dots, R^k.a\}, \langle \min, \max \rangle) \in \Sigma$ then $\Pi_{\varrho^C}(R^I) \subseteq \{ \langle \bar{x} \min \leq \bar{x} \max \rangle \}$ where $\varrho^C = \{R^1.a_1, \dots, R^1.a_n, \dots, R^k.a_1, \dots, R^k.a_m\}$
$\text{MAND} \subseteq \varphi(\varrho) \times (\mathcal{E} \cup \mathcal{V})$	If $\text{MAND}(\{R^1.a_1, \dots, R^1.a_n, \dots, R^k.a_1, \dots, R^k.a_m\}, O) \in \Sigma$ then $O^I \subseteq \Pi_{R^1.a_1} R^I \cup \dots \cup \Pi_{R^1.a_n} R^I \cup \dots \cup \Pi_{R^k.a_1} R^I \cup \dots \cup \Pi_{R^k.a_m} R^I$
$\text{R-SET}_H \subseteq ((\varphi(\varrho) \times \varphi(\varrho)) \times (\varphi(\varrho) \times \varphi(\varrho))) \times (\mu: \varrho \rightarrow \varrho)$ where μ is a partial bijection s.t. for any $\langle \varrho^A, \varrho^B, \mu \rangle \in \text{R-SET}_H$ $\varrho^A = \{R.a \mu(R.a) \in \varrho^B\}$, and $H = \{\text{Sub}, \text{Exc}\}$	<ul style="list-style-type: none"> If $\text{R-SET}_{\text{Sub}}(\{R^1.a_1, \dots, R^1.a_n, \dots, R^k.a_1, \dots, R^k.a_m\}, \{R^1.a, \dots, R^k.a\}, \langle S^1.b_1, \dots, S^1.b_n, \dots, S^k.b_1, \dots, S^k.b_m \rangle, \mu) \in \Sigma$ then $\Pi_{\varrho^A}(R^I) \subseteq \Pi_{\varrho^B}(S^I)$ where $\varrho^A = \{R^1.a_1, \dots, R^1.a_n, \dots, R^k.a_1, \dots, R^k.a_m\}$ and $\varrho^B = \{S^1.b_1, \dots, S^1.b_n, \dots, S^k.b_1, \dots, S^k.b_m\}$ If $\text{R-SET}_{\text{Exc}}(\{R^1.a_1, \dots, R^1.a_n, \dots, R^k.a_1, \dots, R^k.a_m\}, \{R^1.a, \dots, R^k.a\}, \langle S^1.b_1, \dots, S^1.b_n, \dots, S^k.b_1, \dots, S^k.b_m \rangle, \mu) \in \Sigma$ then $\Pi_{\varrho^A}(R^I) \subseteq \Pi_{\varrho^B}(S^I) \cap \Pi_{\varrho^B}(S^I) = \emptyset$ where $\varrho^A = \{R^1.a_1, \dots, R^1.a_n, \dots, R^k.a_1, \dots, R^k.a_m\}$ and $\varrho^B = \{S^1.b_1, \dots, S^1.b_n, \dots, S^k.b_1, \dots, S^k.b_m\}$
$\text{O-SET}_H \subseteq \varphi(\mathcal{E} \cup \mathcal{V}) \times \mathcal{E} \cup \mathcal{V}$ where $H = \{\text{Isa}, \text{Tot}, \text{Ex}\}$	<ul style="list-style-type: none"> If $\text{O-SET}_{\text{Isa}}(\{O_1, \dots, O_n\}, O) \in \Sigma$ then $O_i^I \subseteq O^I$ for $1 \leq i \leq n$ If $\text{O-SET}_{\text{Tot}}(\{O_1, \dots, O_n\}, O) \in \Sigma$ then $O^I \subseteq \bigcup_{i=1}^n O_i^I$ If $\text{O-SET}_{\text{Ex}}(\{O_1, \dots, O_n\}, O) \in \Sigma$ then $\text{O-SET}_{\text{Isa}}(\{O_1, \dots, O_n\}, O) \in \Sigma$ and $O_i^I \cap O_j^I = \emptyset$ for any $1 \leq i < j \leq n$
$\text{O-CARD} \subseteq (\mathcal{E} \cup \mathcal{V}) \times (\mathbb{N} \times (\mathbb{N} \cup \{\infty\}))$	If $\text{O-CARD}(O) = (\min, \max) \in \Sigma$ then $\min \leq \#\{o o \in O^I\} \leq \max$
$\text{R-CARD} \subseteq \mathcal{R} \times (\mathbb{N} \times (\mathbb{N} \cup \{\infty\}))$	If $\text{R-CARD}(R) = (\min, \max) \in \Sigma$ then $\min \leq \#\langle o_1, \dots, o_n \rangle \langle o_1, \dots, o_n \rangle \in R^I \leq \max$
$\text{OBJ} \subseteq \mathcal{R} \times (\mathcal{E} \cup \mathcal{V})$	If $\text{OBJ}(R, O) \in \Sigma$ then $O^I = \text{id}^I(R^I)$
$\text{RING}_J \subseteq \varphi(\varrho \times \varrho)$ where $J = \{\text{Irr}, \text{Asym}, \text{Trans}, \text{Intr}, \text{Antisym}, \text{Acyclic}, \text{Sym}, \text{Ref}\}$	If $\text{RING}_J(R.a, R.b) \in \Sigma$ then $\Pi_{(R.a, R.b)} R^I$ is <i>irreflexive, asymmetric, transitive, intransitive, antisymmetric, acyclic, symmetric, reflexive</i>
$\text{V-VAL}: \mathcal{V} \rightarrow \varphi(\Lambda_D)$ for some $\Lambda_D \in \Lambda$	If $\text{V-VAL}(V) = \{v_1^D, \dots, v_n^D\} \in \Sigma$ then $V^I = \{v_1^D, \dots, v_n^D\}$ for some D

\mathcal{ALCQI} corresponds to the basic DL \mathcal{ALC} equipped with *qualified cardinality restrictions* and *inverse roles*, and it can also be viewed as a fragment of \mathcal{DLR} [4] where relations are restricted to be binary. The difficulty implied by the absence of n -ary relations has been overcome by means of *reification*. Unfortunately, apart from the necessity of introducing reified relations, the restricted expressive power of \mathcal{ALCQI} does not allow to fully capture the semantics of the ORM2 constraints. The analysis of corresponding limitations thus led to identification of a fragment of ORM2, called $\text{ORM2}^{\text{zero}}$, that is maximal with respect to the expressiveness of \mathcal{ALCQI} , and that is still expressive enough to capture the most frequent usage pattern of the modelling community [5].

The $\text{ORM2}^{\text{zero}}$ fragment considers the following constraints:
 $\text{ORM2}^{\text{zero}} = \{\text{TYPE}, \text{FREQ}^-, \text{MAND}, \text{R-SET}^-, \text{O-SET}_{\text{Isa}}, \text{O-SET}_{\text{Tot}}, \text{O-SET}_{\text{Ex}}, \text{OBJ}\}$,
 where: (i) FREQ^- can be applied to only one role at time, and (ii) R-SET^- applies either to a pair of relations of the same arity or to two single roles. The encoding of the semantics of $\text{ORM2}^{\text{zero}}$ is shown in table 3.

Table 3. \mathcal{ALCQI} encoding

Background domain axioms:	$E_i \sqsubseteq \neg(D_1 \sqcup \dots \sqcup D_i)$ for each $i = 1, \dots, n$ $V_i \sqsubseteq D_j$ for each $i = 1, \dots, m$, and some j , with $1 \leq j \leq l$ $D_i \sqsubseteq \prod_{j=i+1}^l \neg D_j$ for each $i = 1, \dots, l$
$\text{TYPE}(R.a, O)$	$\exists \tau(R.a)^- .A_R \sqsubseteq O$
$\text{FREQ}^-(R.a, \langle \min, \max \rangle)$	$\exists \tau(R.a)^- .A_R \sqsubseteq \geq \min \tau(R.a)^- .A_R \sqcap \leq \max \tau(R.a)^- .A_R$
$\text{MAND}(\{R^1.a_1, \dots, R^1.a_n, \dots, R^k.a_1, \dots, R^k.a_m\}, O)$	$O \sqsubseteq \exists \tau(R^1.a_1)^- .A_{R^1} \sqcup \dots \sqcup \exists \tau(R^1.a_n)^- .A_{R^1} \sqcup \dots \sqcup$ $\exists \tau(R^k.a_1)^- .A_{R^k} \sqcup \dots \sqcup \exists \tau(R^k.a_m)^- .A_{R^k}$
• If $A = \{R.a_1, \dots, R.a_n\}$, $B = \{S.b_1, \dots, S.b_n\}$ and $n = \mathcal{Q}_R = \mathcal{Q}_S $:	$\text{R-SET}_{\text{Sub}}^-(A, B) \quad A_R \sqsubseteq A_S$ $\text{R-SET}_{\text{Exc}}^-(A, B) \quad A_R \sqsubseteq A_{\tau_n} \sqcap \neg A_S$
• If $A = \{R.a_i\}$, $B = \{S.b_j\}$:	$\text{R-SET}_{\text{Sub}}^-(A, B) \quad \exists \tau(R.a_i)^- .A_R \sqsubseteq \exists \tau(S.b_j)^- .A_S$ $\text{R-SET}_{\text{Exc}}^-(A, B) \quad \exists \tau(R.a_i)^- .A_R \sqsubseteq A_{\tau_n} \sqcap \neg \exists \tau(S.b_j)^- .A_S$
$\text{O-SET}_{\text{Isa}}(\{O_1, \dots, O_n\}, O)$	$O_1 \sqcup \dots \sqcup O_n \sqsubseteq O$
$\text{O-SET}_{\text{Tot}}(\{O_1, \dots, O_n\}, O)$	$O \sqsubseteq O_1 \sqcup \dots \sqcup O_n$
$\text{O-SET}_{\text{Ex}}(\{O_1, \dots, O_n\}, O)$	$O_1 \sqcup \dots \sqcup O_n \sqsubseteq O$ $O_i \sqsubseteq \prod_{j=i+1}^n \neg O_j$ for each $i = 1, \dots, n$
$\text{OBJ}(R, O)$	$O \equiv A_R$

Given the encoding above, a fragment of \mathcal{ALCQI} KB corresponding to the schema from the Fig. 1 is the following (where reified relations have been prefixed with ‘R-’):

Example 1. $\exists \tau(\text{reportsTo.sub})^- .\text{R-reportsTo} \sqsubseteq \text{Admin}$
 $\exists \tau(\text{reportsTo.obj})^- .\text{R-reportsTo} \sqsubseteq \text{AreaManager}$
 $\text{Admin} \sqsubseteq \exists \tau(\text{reportsTo.sub})^- .\text{R-reportsTo}$

The correctness of the introduced encoding is guaranteed by the following theorem:

Theorem 1. *Let Σ^{zero} be an $\text{ORM2}^{\text{zero}}$ conceptual schema and $\Sigma^{\mathcal{ALCQI}}$ the \mathcal{ALCQI} knowledge base constructed as described above. Then an entity/value type O is consistent in Σ^{zero} if and only if the concept O is satisfiable w.r.t. $\Sigma^{\mathcal{ALCQI}}$.*

5 Prototype of Automated Reasoning Support Tool

The ORM2 automated reasoning support tool is implemented in Java and includes a parser for ORM2 linear syntax, a set of Java classes representing the ORM2 knowledge database, a translator into an OWL2 ontology and a modal reasoning engine using HerMiT or FaCT++ as an underlying reasoner. The graphical user interface of a tool is introduced on Figure 2 and contains controls which allow to select an input file, underlying reasoner, output file and output format for resulting ontology (if needed). The consistency check for an input ORM2 schema is performed after loading the input file and the result of the check is communicated by visual flag as well as by a detailed log in the corresponding window.

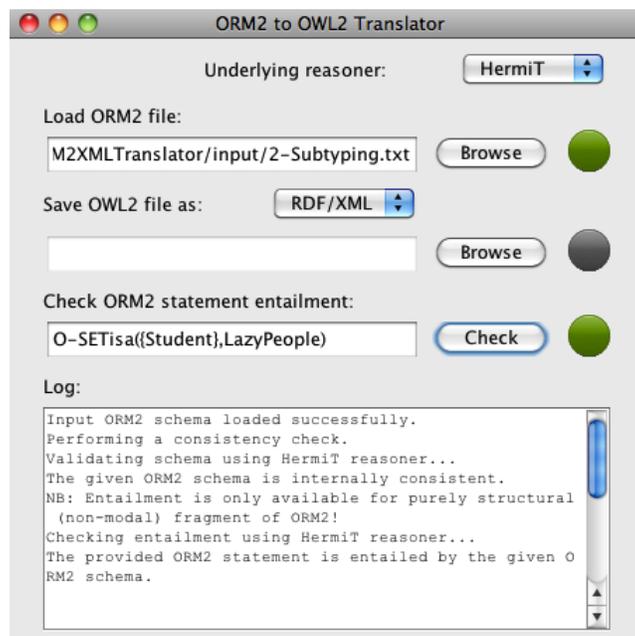


Fig. 2. A conceptual schema including an instantiation of most of the ORM2 constraints

Let us now consider the following example in ORM2 linear syntax. This model describes a domain of university personnel, which is partitioned into three mutually exclusive categories: research staff, administration and some lazy people. We then introduce an entity type Student as a part of university personnel, which is claimed to be neither researchers nor administration staff.

```

ENTITYTYPES: {UNI-Personnel,LazyPeople,Student,Admin,RTStaff}
O-SETtot({LazyPeople,RTStaff,Admin},UNI-Personnel)
O-SETex({LazyPeople,RTStaff,Admin},UNI-Personnel)
O-SETisa({Student},UNI-Personnel)
O-SETex({Student,RTStaff},UNI-Personnel)
O-SETex({Student,Admin},UNI-Personnel)

```

The resulting encoding in \mathcal{ALCQI} will look like the following:

$$\begin{aligned} \text{UNI-Personnel} &\sqsubseteq \text{RTStaff} \sqcup \text{Admin} \sqcup \text{LazyPeople} \\ \text{RTStaff} &\sqsubseteq \neg \text{Admin} \sqcup \neg \text{LazyPeople} \\ \text{Admin} &\sqsubseteq \neg \text{LazyPeople} \\ \text{Student} &\sqsubseteq \neg \text{RTStaff} \sqcup \neg \text{Admin} \\ \text{Student} &\sqsubseteq \text{UNI-Personnel} \end{aligned}$$

After loading the linear syntax input file, the corresponding conceptual schema is automatically checked for consistency (see Figure 2). As a result of the consistency check we obtain the message 'The given ORM2 schema is internally consistent', which confirms the correctness of the schema.

Implemented entailment check functionality allows user to analyze the conceptual schema and to discover interesting inferences. For example, let us analyze the entity type Student, which is claimed to be part of the university personnel. However, since the partition of UNI-Personnel described above is total, the only possibility for Student to be non-empty is to be equivalent to the entity type LazyPeople. Which is indeed confirmed by the inference engine of the implemented prototype when performing entailment check for the following ORM2 statement (see Figure 2):

$$\text{O-SETisa}(\{\text{Student}\}, \text{LazyPeople})$$

6 Related Works

In the last few years, several papers addressed the issue of encoding ORM2 conceptual schema into DL knowledge bases [15,14,13,11]. Among those proposals, [15] can be taken as the only one going through the encoding with a formal perspective. In particular, [15] pretends to start from the Halpin's FOL semantics, and introduces an encoding of a fragment of ORM2 into the logic \mathcal{DLR}_{ifd} [3]. In general, the paper suffers from the presence of several imprecisions, redundancy, and syntactical mistakes that makes the proposed mapping solutions not always clearly understandable. Moreover, the bottom-up approach that avoids the specification of a complete theoretical framework for the mapping of the ORM2 semantics into \mathcal{DLR}_{ifd} , makes some of these solutions extremely questionable, such as in the case where 'objectification' is simply treated as 'relation reification' in DL.

As regards to [14], we mostly rely here on the extensive review already made by Keet in [15]. Starting from this, it should be also noticed that subsequent attempts, focused on the possibilities of encoding ORM2 into the the web ontology language OWL2 [13,11], suffer from the same formal inconsistencies and limitations of [14]. In particular, [14] is misleading with respect to the underlying DL formalism: distinct DL languages (e.g. \mathcal{DLR} , plus $\mathcal{DLR-Lite}$, plus \mathcal{SROIQ} , plus 'role composition' operator) are there arbitrary mixed together. No special semantics is provided by [14] in correspondence with these combinations, nor theorems showing the complexity of reasoning with them.

Another paper focused on the encoding of ORM2 in OWL has also been recently published [16]. The paper introduces a set of informal 'rules' devoted to the mapping of a subset of ORM2 constructs into OWL Manchester Syntax [7]. Unfortunately, the paper is misleading in several respects (for instance: (i) the OWL EquivalentTo, instead of

the `SubClassOf`, is erroneously introduced several times; (ii) optionality of uniqueness constraints is definitively lost). In general, the paper covers a fragment that is smaller than $\text{ORM2}^{\text{zero}}$, and the proposed mapping mostly remains formally unjustified.

7 Concluding Remarks

In this paper we introduced a linear syntax and a complete set-theoretic semantics for the ORM2 conceptual modelling language. A decidable, and computationally tractable, fragment of ORM2 has been clearly identified and mapped into the DL logic \mathcal{ALCQI} . Finally, a first reasoning support prototype for ORM2 has been implemented, which enables consistency and entailment checks for the defined fragment of ORM2. Future theoretic works will be mainly focused on the extension of the $\text{ORM2}^{\text{zero}}$ towards the identification of a more expressive, still decidable, ‘object role’ modelling language. The practical objectives of the research will be directed towards full integration of the prototype into third-party solutions providing graphical user interface for designing ORM2 conceptual schemas (e.g. NORMA plugin for Microsoft Visual Studio). In particular, such integration will benefit from reasoning capabilities of the tool by providing the user with a list of all meaningful inferences entailed by the original schema.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The description logic handbook: theory, implementation, and applications. Cambridge University Press, New York, NY, USA (2003)
2. Berardi, D., Cali, A., Calvanese, D., Giacomo, G.D.: Reasoning on UML class diagrams. *Artificial intelligence* 168 (2003)
3. Calvanese, D., De Giacomo, G., Lenzerini, M.: Identification constraints and functional dependencies in description logics. In: *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 1*. pp. 155–160. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001), <http://dl.acm.org/citation.cfm?id=1642090.1642111>
4. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Description logic framework for information integration. In: *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*. pp. 2–13 (1998)
5. Calvanese, D., Lenzerini, M., Nardi, D.: Description logics for conceptual data modeling. In: Chomicki, J., Saake, G. (eds.) *Logics for Databases and Information Systems*. pp. 229–263. Kluwer (1998)
6. Ceravolo, P., Fugazza, C., Leida, M.: Modeling semantics of business rules. In: *Proceedings of the Inaugural IEEE International Conference On Digital Ecosystems and Technologies (IEEE-DEST) (February 2007)*
7. Grau, B.C., Hitzler, P., Shankey, C., Wallace, E. (eds.): *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions*, Athens, Georgia, USA, November 10-11, 2006, CEUR Workshop Proceedings, vol. 216. CEUR-WS.org (2006)
8. Haarslev, V., Miller, R.: Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In: *KR'00*. pp. 273–284 (2000)
9. Halpin, T.: *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD thesis, Department of Computer Science, University of Queensland (1989)

10. Halpin, T., Morgan, T.: Information modeling and relational databases: from conceptual analysis to logical design. Morgan Kaufmann, 2nd edn. (2001)
11. Hodrob, R., Jarrar, M.: ORM to OWL2 DL mapping. In: International conference on intelligent semantic web: applications and services. ACM (2010)
12. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning, pp. 161–180. LPAR '99, Springer-Verlag, London, UK (1999), <http://dl.acm.org/citation.cfm?id=645709.664314>
13. Jarrar, M.: Mapping ORM into the SHOIN/OWL description logic – towards a methodological and expressive graphical notation for ontology engineering. In: OTM 2007 workshops: Proceedings of the International Workshop on Object-Role Modeling (ORM'07). LNCS, vol. 4805, pp. 729–741. Springer (November 2007)
14. Jarrar, M.: Towards automated reasoning on ORM schemes. mapping ORM into the \mathcal{DLR}_{ifd} description logic. In: Proceedings of the 26th International Conference on Conceptual Modeling (ER 2007). LNCS, vol. 4801, pp. 181–197. Springer (November 2007)
15. Keet, M.: Mapping the Object-Role Modeling language ORM2 into description logic language \mathcal{DLR}_{ifd} . Tech. Rep. KRDB07-2, KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano (2007)
16. Wagih, H.M., ElZanfaly, D.S., Kouta, M.M.: Mapping Object Role Modeling 2 schemes to OWL2 ontologies. In: Ting, Z. (ed.) Proceedings of the 3rd IEEE International Conference on Computer Research and Development (ICCRD), Shanghai, China, March 11-13, 2011, vol. 4, pp. 126–132. IEEE Press (2011)

Adding Context to Tableaux for DLs

Weili Fu and Rafael Peñaloza

Theoretical Computer Science,
TU Dresden, Germany

`weili.fu77@googlemail.com`, `penaloza@tcs.inf.tu-dresden.de`

Abstract. We consider the problem of reasoning with ontologies where every axiom is associated to a context, and contexts are related through a total order. These contexts could represent, for example, a degree of trust associated to the axiom, or a level of granularity for the knowledge provided. We describe an extension of tableaux-based decision procedures into methods that compute the best-fitting context for the consequences of an ontology, and apply it to the tableaux algorithm for \mathcal{ALC} . We also describe an execution strategy that preserves most of the standard optimizations used in modern DL reasoners.

1 Introduction

Several non-standard reasoning tasks for DL ontologies can be described as repeated standard reasoning, over some of its sub-ontologies. This is the case, for example, in axiom-pinpointing [14,12,3], where the task is to identify the class of sub-ontologies entailing a consequence, or access control [2], where users are assigned views to specific subontologies, and one wants to decide which users can or cannot deduce some implicit consequence.

One can think of the sub-ontologies over which standard reasoning is performed as contextual views to the whole ontology. The task is then to partition the contexts into those that entail and those that do not entail the desired consequence. An obvious way to solve this task is to test each of the potentially exponentially many sub-ontologies for the consequence, and classify them according to the result. However, it is possible to exploit the structure of the set of contexts to obtain a more efficient method.

Notice that every set of sub-ontologies defines a partial ordering \leq via the superset relationship: $\mathcal{O} \leq \mathcal{O}'$ iff $\mathcal{O}' \subseteq \mathcal{O}$. This partial order can always be extended to a lattice in which every context is join-prime [6]. It has been shown that there is an element ν in this lattice, called the *boundary*, such that the following holds for every context \mathcal{O} represented by a label $\ell_{\mathcal{O}}$: \mathcal{O} entails the consequence iff $\ell_{\mathcal{O}} \leq \nu$ [2]. If the lattice is distributive, then the boundary can be computed in polynomial time on the size of the input ontology [3].

So far, all the methods implemented for computing boundaries are based on a black-box approach, in which an unmodified reasoner is called repeatedly until the boundary has been found. In fact, attempts to produce a glass-box approach for e.g. axiom-pinpointing [4], where the reasoner is modified to directly compute

the boundary, encounter the problem that most of the optimizations used in implemented DL reasoners do not work in the modified procedures, making the glass-box approaches much less efficient than black-box ones. Moreover, glass-box extensions are not even guaranteed to terminate.

In this paper we focus on a special case in which the contexts are linearly ordered. That is, for every pair of sub-ontologies $\mathcal{O}, \mathcal{O}'$, either $\mathcal{O} \subseteq \mathcal{O}'$ or $\mathcal{O}' \subseteq \mathcal{O}$. This simple setting still covers a wide variety of applications, such as possibilistic reasoning [13,11], trust management, or granularity reasoning. We develop a glass-box approach for extending tableau-based decision algorithms into procedures that compute the boundary for the consequence decided by the original tableaux. A prioritization ordering of the rule applications of these procedures ensures that the boundary-computation procedure behaves almost identically to the original tableau. This entails not only termination of the algorithm, but also that most of the optimization techniques used by modern DL reasoners are still applicable to the extensions. An additional benefit of our method is that it can deal with arbitrary blocking conditions, without losing its correctness.

The paper is divided as follows. We first introduce the basic reasoning problems we are interested in, followed by a general notion of tableaux. In Section 4 we describe the labeled extensions of tableaux, that output a boundary for the desired property, and provide the prioritization ordering that ensures that these extensions behave well. We wrap up by arguing that our approach can be seen as a special case of incremental reasoning, and hence should be easy to implement in modern DL reasoners.

2 Basic Definitions

We start by introducing the general notion of consequence properties that are decided by general tableaux. To avoid unnecessary confusion, we present slightly simplified versions of the notions of consequence properties and tableaux, which suffice for the goals of this paper. We consider also a general notion of axioms, which can e.g. be assertions, GCIs, or concept definitions.

Definition 1. *Let \mathfrak{D} be a set of axioms, and let $\mathcal{P}_{fin}(\mathfrak{D})$ denote the set of all finite subsets of \mathfrak{D} . A consequence property (c-property) is a set $\mathcal{P} \subseteq \mathcal{P}_{fin}(\mathfrak{D})$ such that $\mathcal{O} \in \mathcal{P}$ implies $\mathcal{O}' \in \mathcal{P}$ for every $\mathcal{O} \subseteq \mathcal{O}'$.*

Intuitively, c-properties model consequence relations in logic that are monotonic; i.e. they describe which sets of axioms have or entail a desired consequence. For example, one can consider the set \mathfrak{D} of all concept- and role-assertions in the DL \mathcal{ALC} and \mathcal{P}_{exa} the set of all *inconsistent* \mathcal{ALC} ABoxes. If \mathcal{O}_{exa} is the ABox having the assertions

$$\text{ax}_1 : \neg\exists r.B(a) \quad \text{ax}_2 : (\neg A \sqcap B)(b) \quad \text{ax}_3 : \forall r.A(a) \quad \text{ax}_4 : r(a, b) \quad (1)$$

then $\mathcal{O}_{\text{exa}} \in \mathcal{P}_{\text{exa}}$; that is, \mathcal{O}_{exa} is inconsistent. In general, we will call the sets $\mathcal{O} \in \mathcal{P}_{fin}(\mathfrak{D})$ *ontologies*.

For several applications, axioms cannot be considered to have all equal importance, but are provided with a label that represents the context to which they belong. We consider only labels that come from a finite totally ordered set (L, \leq) . For the rest of this paper, we will denote the elements of L as $\ell_0, \ell_1, \dots, \ell_k$, with the implicit ordering $\ell_i \leq \ell_j$ iff $i \leq j$. In particular, ℓ_0 is the least- and ℓ_k is the greatest-element of L .

We will use the elements of the set L to define different *contexts* of an ontology. Depending on the specific application at hand, these contexts can have different meanings, such as access rights, level of expertise, trustworthiness, necessity degree, etc.

Every axiom t in an ontology \mathcal{O} is assigned a label $\text{lab}(t) \in L$, which expresses the contexts from which this axiom can be accessed. An ontology extended with such a labeling function lab will be called a *labelled ontology*. Each element $\ell \in L$ defines the context sub-ontology

$$\mathcal{O}_{\geq \ell} := \{t \in \mathcal{O} \mid \text{lab}(t) \geq \ell\}.$$

Clearly, if $\ell \leq \ell'$, then $\mathcal{O}_{\geq \ell'} \subseteq \mathcal{O}_{\geq \ell}$. Conversely, every ontology \mathcal{O} defines an element $\lambda_{\mathcal{O}} \in L$ (called the *label* of \mathcal{O}), given by $\lambda_{\mathcal{O}} := \min\{\text{lab}(t) \mid t \in \mathcal{O}\}$. It follows from this definition that $\lambda_{\mathcal{O}} \leq \lambda_{\mathcal{O}'}$ whenever $\mathcal{O}' \subseteq \mathcal{O}$.

For a given labeled ontology, we want to compute the so-called *boundary* of the c-property \mathcal{P} . Intuitively, the boundary divides the contexts where \mathcal{P} follows from those where it does not follow.

Definition 2. *Let \mathcal{O} be a labeled ontology and \mathcal{P} a c-property. An element $\nu \in L$ is called a $(\mathcal{O}, \mathcal{P})$ -boundary if for every element $\ell \in L$ the following holds:*

$$\ell \leq \nu \quad \text{iff} \quad \mathcal{O}_{\geq \ell} \in \mathcal{P}.$$

When it is clear from the context, we will usually omit the prefix $(\mathcal{O}, \mathcal{P})$, and call this ν a boundary.

Continuing with our example, let the label of every axiom appearing in (1) be $\text{lab}(\text{ax}_i) = \ell_i, 1 \leq i \leq 4$. The $(\mathcal{O}_{\text{exa}}, \mathcal{P}_{\text{exa}})$ -boundary is then ℓ_2 since the sub-ontology $\mathcal{O}_{\geq \ell_2} = \{\neg A \sqcap B(b), \forall r.A(a), r(a, b)\}$ is inconsistent, while the ontology $\mathcal{O}_{\geq \ell_3} = \{\forall r.A(a), r(a, b)\}$ is consistent, i.e. does not belong to \mathcal{P}_{exa} .

It was shown in [2] that if $\mathcal{O} \in \mathcal{P}$, then the $(\mathcal{O}, \mathcal{P})$ -boundary always exists, is unique, and can be computed using axiom-pinpointing techniques. Given an ontology \mathcal{O} that belongs to a c-property \mathcal{P} , a minimal (w.r.t. set inclusion) sub-ontology $\mathcal{O}' \subseteq \mathcal{O}$ that still belongs to \mathcal{P} is called a *MinA* for \mathcal{O}, \mathcal{P} .¹ If $\mathcal{O}_1, \dots, \mathcal{O}_m$ are all the MinAs for \mathcal{O}, \mathcal{P} , then

$$\nu = \max\{\lambda_{\mathcal{O}_i} \mid 1 \leq i \leq m\} \tag{2}$$

is the $(\mathcal{O}, \mathcal{P})$ -boundary. However, using this method for computing the boundary may result in an unnecessary overhead. In our running example, we can see that $\{\text{ax}_1, \text{ax}_2, \text{ax}_4\}$ and $\{\text{ax}_2, \text{ax}_3, \text{ax}_4\}$ are the MinAs for $\mathcal{O}_{\text{exa}}, \mathcal{P}_{\text{exa}}$. The second

¹ These minimal sub-ontologies are also called *justifications* [9].

MinA has label ℓ_2 . If we happen to compute this second MinA first, then there is no need to continue computing MinAs, as it is clear that any new MinA \mathcal{O}' must contain the axiom ax_1 , and hence have label $\ell_1 < \ell_2$. This new MinA will have no influence in the computation of the maximum from Equation (2). This problem is in fact more pronounced than what the example shows, since a single ontology can have exponentially many MinAs [5], and the boundary can be computed by a black-box algorithm that calls a decision procedure for \mathcal{P} at most $k = |L|$ times; namely, once for each $\ell \in L$, to decide whether or not $\mathcal{O}_{\geq \ell} \in \mathcal{P}$.

3 General Tableaux

In this section we recall some of the notions of general tableaux [4]. We use \mathcal{V} and \mathcal{D} to denote two disjoint, countably infinite sets of *variables* and *constants*, respectively. A *signature* Σ is a set of predicate symbols, where each predicate $P \in \Sigma$ is equipped with an arity. A Σ -*assertion* is of the form $P(a_1, \dots, a_n)$ where $P \in \Sigma$ is an n -ary predicate and $a_1, \dots, a_n \in \mathcal{D}$. Similarly, a Σ -*pattern* is of the form $P(x_1, \dots, x_n)$ where $P \in \Sigma$ and $x_1, \dots, x_n \in \mathcal{V}$. Whenever the signature is clear from the context, we simply say pattern (assertion). Given a set of assertions A (resp. patterns B), $\text{cons}(A)$ (resp. $\text{var}(B)$) denotes the set of constants (resp. variables) occurring in A (resp. B).

A *substitution* is a mapping $\sigma : V \rightarrow \mathcal{D}$, where $V \in \mathcal{P}_{fin}(\mathcal{V})$. In this case, we say that σ is a *substitution on V* . The substitution θ on V' *extends* σ on V if $V \subseteq V'$ and $\theta(x) = \sigma(x)$ for all $x \in V$. If B is a set of patterns with $\text{var}(B) \subseteq V$, then $B\sigma$ denotes the set of assertions obtained from B by replacing each variable by its image over the substitution σ .

Definition 3. Let \mathfrak{D} be a set of axioms. A tableau for \mathfrak{D} is a tuple $S = (\Sigma, \mathcal{R}, \mathcal{C})$ where

- Σ is a signature,
- \mathcal{R} is a set of expansion rules of the form $(B_0, \mathcal{N}) \rightarrow \{B_1, \dots, B_m\}$, where B_0, \dots, B_m are finite sets of Σ -patterns and $\mathcal{N} \in \mathcal{P}_{fin}(\mathfrak{D})$, and input rules of the form $t \rightarrow B$, where $t \in \mathfrak{D}$ and B is a finite set of Σ -assertions, and
- \mathcal{C} is a set of finite sets of Σ -patterns, called clashes.

Given an expansion rule $R : (B_0, \mathcal{N}) \rightarrow \{B_1, \dots, B_m\}$, the variable y is a *fresh variable* in R if it occurs in one of the sets B_1, \dots, B_m but not in B_0 .

An S -state is a pair $\mathfrak{S} = (A, \mathcal{O})$ where A is a finite set of assertions and \mathcal{O} is a finite set of axioms. The tableau algorithm works on sets of S -states. It starts with the set $\mathcal{M} = \{(\emptyset, \mathcal{O})\}$ and uses the rules in \mathcal{R} to modify this set. Each rule application picks an S -state \mathfrak{S} from \mathcal{M} and replaces it by finitely many new S -states that extend the first component of \mathfrak{S} . When no rules are applicable, then it tests whether the resulting set of S -states contains a clash; in that case, it accepts the ontology, and rejects it otherwise.

Definition 4. An input rule $t \rightarrow B$ is applicable to an S -state (A, \mathcal{O}) if $t \in \mathcal{O}$ and $B \not\subseteq A$. An expansion rule $R : (B_0, \mathcal{N}) \rightarrow \{B_1, \dots, B_m\}$ is applicable to an

S -state (A, \mathcal{O}) with substitution ρ on $\text{var}(B_0)$ if (i) $\mathcal{N} \subseteq \mathcal{O}$, (ii) $B_0\rho \subseteq A$, and (iii) for every $1 \leq i \leq m$ and every substitution ρ' on $\text{var}(B_0 \cup B_i)$ extending ρ we have $B_i\rho' \not\subseteq A$.

Given a set of S -states \mathcal{M} , application of an input rule R to $\mathfrak{S} = (A, \mathcal{O}) \in \mathcal{M}$ yields the set $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B, \mathcal{O})\}$; application of an expansion rule R to $\mathfrak{S} = (A, \mathcal{O}) \in \mathcal{M}$ with ρ yields $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B_i\sigma, \mathcal{O}) \mid 1 \leq i \leq m\}$, where σ is a substitution that extends ρ and maps the fresh variables of R to distinct new constants. If \mathcal{M}' is obtained from \mathcal{M} by a rule application, we write $\mathcal{M} \rightarrow_S \mathcal{M}'$. \mathcal{M} is saturated if there is no \mathcal{M}' with $\mathcal{M} \rightarrow_S \mathcal{M}'$.

The S -state (A, \mathcal{O}) contains a clash if there is a $C \in \mathcal{C}$ and a substitution ρ on $\text{var}(C)$ with $C\rho \subseteq A$. \mathcal{M} is full of clashes if every $\mathfrak{S} \in \mathcal{M}$ contains a clash.

A simple example of a tableau is the one for deciding inconsistency of \mathcal{ALC} ABoxes. Its clashes are all the sets $\{D, \neg D\}$, where D is a concept name. It has different expansion rules dealing with the constructors of \mathcal{ALC} . For example, the rules for existential restrictions and disjunction are as follows:

$$\begin{aligned} R_{\exists} &: (\{\exists r.C(x)\}, \emptyset) \rightarrow \{\{r(x, y), C(y)\}\}, \\ R_{\sqcup} &: (\{C \sqcup D(x)\}, \emptyset) \rightarrow \{\{C(x)\}, \{D(x)\}\}. \end{aligned}$$

It also has input rules, for dealing with the ABox axioms used. For example, for concept assertions, we use

$$R_a : C(a) \rightarrow \{C(\bar{a})\},$$

where \bar{a} is the constant representing the individual name a .

A tableau is *correct* for a c-property \mathcal{P} if every chain of rule applications starting with $\mathcal{M}_0 = \{(\emptyset, \mathcal{O})\}$ eventually reaches a saturated set of S -states \mathcal{M} (i.e. it terminates) and it holds that $\mathcal{O} \in \mathcal{P}$ iff \mathcal{M} is full of clashes.

In general, tableaux are not guaranteed to terminate, as their expansion rules may trigger the applicability of new expansion rules. This happens, for instance, in the tableaux algorithm for deciding inconsistency w.r.t. general TBoxes. To avoid this problem, a *blocking condition* is usually introduced. Intuitively, blocking disallows the application of a rule, whenever its application would not lead to the production of any new clashes; that is, when further expansions would not change the acceptance status of the input ontology. Several blocking conditions—like e.g. subset blocking [1], equality blocking [7], pairwise blocking [8], etc—have been studied in the literature. Rather than trying to define and deal with each of these conditions independently, we consider a general notion of blocking.

A *blocking condition* is simply a set of finite sets of Σ -assertions and Σ -patterns, using only the variables from $\text{var}(B_0)$. Intuitively, these describe the situations in which the rule should not be applied. Every expansion rule is then extended with a blocking condition \mathcal{B} , and the rule applicability condition is restricted to satisfy additionally (iv) $A \notin \mathcal{B}\rho$. The notions of saturated sets of S -states and correctness w.r.t. a c-property are adapted accordingly. For example, the subset blocking condition disallowing the applicability of the existential rule in \mathcal{ALC} is described by the set of all finite sets of Σ -assertion and Σ -patterns B

where the only variable used is x and such that there exist constants a_1, \dots, a_n and roles r_1, \dots, r_n such that $r_i(a_i, a_i + 1)$ for all $i, 1 \leq i < n$, $r_i(a_n, x)$, and $\{C \mid C(x) \in B\} \subseteq \{C \mid C(a_1) \in B\}$.

In the next section, we show how to transform tableaux that are correct for a c-property \mathcal{P} into procedures that directly compute the $(\mathcal{O}, \mathcal{P})$ -behaviour, for any given ontology \mathcal{O} . We focus first on tableaux without any blocking condition, which can be used for deciding consistency of \mathcal{ALC} ABoxes. Later on, we show how to extend this construction to tableaux with blocking conditions, allowing us to reason w.r.t. general TBoxes also.

4 Boundary Extensions of General Tableaux

Given a tableau S that is correct for the c-property \mathcal{P} , we show how to construct an algorithm that computes the boundary for \mathcal{P} . For the moment, we focus on tableaux without blocking conditions, but later describe how to deal with blocking also.

For an input labeled ontology \mathcal{O} , the modified algorithm also works on sets of S -states, but every assertion a occurring in the first component of an S -state is also equipped with a label $\text{lab}(a)$; we call this a *labeled S -state*. The application of rules must take the labels of the assertions and axioms into account.

If A is a set of labeled assertions and $\ell \in L$, then we say that an assertion a is *ℓ -insertable into A* if either (i) $a \notin A$ or (ii) $a \in A$ but $\ell > \text{lab}(a)$. Given a set B of (unlabeled) assertions and a set A of labeled assertions, the set of *ℓ -insertable elements of B into A* is

$$\text{ins}_\ell(B, A) := \{b \in B \mid b \text{ is } \ell\text{-insertable into } A\}.$$

The ℓ -insertion of these elements into A yields the set of labeled assertions $A \uplus_\ell B := A \cup \text{ins}_\ell(B, A)$, where each assertion $a \in A \setminus \text{ins}_\ell(B, A)$ keeps its old label $\text{lab}(a)$, each assertion in $\text{ins}_\ell(B, A) \setminus A$ is labeled with ℓ , and the label of each assertion $b \in A \cap \text{ins}_\ell(B, A)$ is updated to ℓ .

Definition 5. *The input rule $t \rightarrow B$ is labeled applicable to a labeled S -state $\mathfrak{S} = (A, \mathcal{O})$ if $t \in \mathcal{O}$ and $\text{ins}_{\text{lab}(t)}(B, A) \neq \emptyset$. An expansion rule of the form $R : (B_0, \mathcal{N}) \rightarrow \{B_1, \dots, B_m\}$ is labeled applicable to a labeled S -state (A, \mathcal{O}) with substitution ρ on $\text{var}(B_0)$ if (i) $\mathcal{N} \subseteq \mathcal{O}$, (ii) $B_0\rho \subseteq A$, and (iii) for every $1 \leq i \leq m$ and every substitution ρ' on $\text{var}(B_0 \cup B_i)$ extending ρ we have $\text{ins}_\ell(B_i\rho', A) \neq \emptyset$, where $\ell := \min\{\text{lab}(\alpha) \mid \alpha = b\rho, b \in B_0 \text{ or } \alpha = s, s \in \mathcal{N}\}$. We call this ℓ the degree of the rule application.*

Given a set of labeled S -states \mathcal{M} , the labeled application of the input rule R to $\mathfrak{S} = (A, \mathcal{O}) \in \mathcal{M}$ yields the set $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \uplus_{\text{lab}(t)} B, \mathcal{O})\}$; labeled application of the expansion rule R to $\mathfrak{S} = (A, \mathcal{O}) \in \mathcal{M}$ with ρ yields the new set $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \uplus_\ell B_i\sigma, \mathcal{O}) \mid 1 \leq i \leq m\}$, where ℓ is the degree of the rule application, defined above, and σ is a substitution that extends ρ and maps the fresh variables of R to distinct new constants. If \mathcal{M}' is obtained from \mathcal{M} by a labeled rule application, we write $\mathcal{M} \rightarrow_{S^{\text{lab}}} \mathcal{M}'$. \mathcal{M} is labeled saturated if there is no \mathcal{M}' with $\mathcal{M} \rightarrow_{S^{\text{lab}}} \mathcal{M}'$.

Consider a finite chain of labeled rule applications $\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\text{lab}}} \mathcal{M}$ such that \mathcal{M} is labeled saturated. The label of an assertion appearing in \mathcal{M} expresses the contexts that can derive this assertion. A clash in an S -state depends on the joint presence of possibly several assertions; thus, we need to find the contexts from which all of these assertions can be derived: this is given by the minimum of the labels of all these assertions. To decide the property \mathcal{P} , it suffices to have one clash per S -state \mathfrak{S} ; hence, we are only interested in the *maximum* of the labels of all the clashes in a given state. As every S -state in \mathcal{M} is required to have at least one clash, we again compute the minimum of the labels obtained from each S -state. We formalize this next.

Definition 6. *A set of assertions A' is called a clash set in a labeled S -state $\mathfrak{S} = (A, \mathcal{O})$ if there is a clash $C \in \mathcal{C}$ and a substitution ρ on $\text{var}(C)$ with $A' = C\rho$. The label of this clash set is $\ell_{A'} = \min\{\text{lab}(a) \mid a \in A'\}$.*

Given a set of labeled S -states $\mathcal{M} = \{\mathfrak{S}_1, \dots, \mathfrak{S}_n\}$, the clash degree induced by \mathcal{M} is $\ell_{\mathcal{M}} := \min\{\max\{\ell_{A'} \mid A' \text{ is a clash set in } \mathfrak{S}_i\} \mid 1 \leq i \leq n\}$

It can be shown, using similar techniques to the ones developed in [4] for pinpointing extensions of general tableaux that the clash degree is always the boundary for the c -property decided by the original tableau.

Theorem 7. *Let \mathcal{P} be a c -property on \mathfrak{D} and S a correct tableau for \mathcal{P} . For every ontology $\mathcal{O} \in \mathcal{P}_{\text{fin}}(\mathfrak{D})$ the following holds:*

for every finite chain of rule applications $\{(\emptyset, \mathcal{O})\} \xrightarrow{}_{S^{\text{lab}}} \mathcal{M}$ with \mathcal{M} labeled saturated, the clash degree $\ell_{\mathcal{M}}$ induced by \mathcal{M} is a $(\mathcal{O}, \mathcal{P})$ -boundary.*

Notice that this result does not depend on the order in which rules are applied. From a given set of S -states, several rules could be applicable, but the correctness of the labeled extension does not depend on which one is chosen first. This is an important feature of tableaux and their extensions, as it allows optimizations based on the choice of an ordering that leads to shorter chains of rule applications.

Unfortunately, this general approach for extending tableaux into boundary-computation methods also inherits some of the negative properties of pinpointing extensions of tableaux. In particular, (i) the labeled extension of a terminating tableau is not guaranteed to terminate (see [4] for an example), and (ii) even if it terminates, the labeled extension needs to run until saturation to guarantee that the clash label computed is indeed the boundary. The second point is used as an argument against glass-box approaches, as it disallows one of the most basic optimizations for tableaux, namely stopping the application of expansion rules on states where a clash has already been detected.

We now show that if we prioritize rule applications with a higher degree, then both of these problems disappear. Moreover, the same approach will allow us to deal with general blocking conditions in Section 4.2.

4.1 Prioritized Rule Applications

By definition, the boundary is the largest context from which the c-property can be derived. A simple idea to compute this boundary is then to try to first produce all the assertions that can be derived from a context—that is, from all the axioms having a label greater or equal to some ℓ_i —before applying rules that use axioms with a smaller label. We implement this idea by prioritizing labeled rule applications with a higher degree.

Definition 8. *The ordered extension of a tableau S is the labeled extension of S where, if several rules are applicable to a set of labeled S -states \mathcal{M} , then one rule application having the highest degree is applied.*

We write $\mathcal{M} \rightarrow_{S^{\text{ord}}} \mathcal{M}'$ if \mathcal{M}' is obtained from \mathcal{M} by an ordered rule application. \mathcal{M} is ordered saturated if there is no \mathcal{M}' with $\mathcal{M} \rightarrow_{S^{\text{ord}}} \mathcal{M}'$.

Several rule applications may have the same (highest) degree. In that case, it is irrelevant which one of these is chosen. This prioritization ordering has the consequence that the degree of successive rule applications is non-increasing.

Lemma 9. *Let $\mathcal{M}_0 \rightarrow_{S^{\text{ord}}} \mathcal{M}_1 \rightarrow_{S^{\text{ord}}} \mathcal{M}_2$. If \mathcal{M}_2 is obtained by applying a rule with degree ℓ to \mathcal{M}_1 , and \mathcal{M}_1 is obtained by applying a rule with degree ℓ' to \mathcal{M}_0 , then $\ell \leq \ell'$.*

Proof. \mathcal{M}_1 differs from \mathcal{M}_0 by having some S -states $\mathfrak{S}_1, \dots, \mathfrak{S}_m$ that modify the assertional component of a S -state $\mathfrak{S} \in \mathcal{M}_0$ in the following way: some new assertions labeled with ℓ' are added, and some existing assertions get their label increased to ℓ' . In any case, all assertions appearing in some S -state of \mathcal{M}_1 with label greater than ℓ' are also in an S -state of \mathcal{M}_0 . If $\ell > \ell'$, then all the assertions and axioms used to execute the rule $\mathcal{M}_1 \rightarrow_{S^{\text{ord}}} \mathcal{M}_2$ have a label strictly greater than ℓ' , and hence this rule was applicable also to an S -state in \mathcal{M}_0 . But $\ell > \ell'$ implies that ℓ' would not be applied, since it violates the prioritization ordering, hence $\ell \leq \ell'$. \square

A simple consequence of this lemma is that the labels of assertions appearing in the S -states are never modified by further ordered rule applications, as such modification would imply a rule application with a higher degree than the rule that originally created the assertion. This means that a rule is only ordered applicable if it adds new assertions to the S -state, and thus, for any set of S -states reachable from the initial set $\{(\emptyset, \mathcal{O})\}$ a rule is ordered applicable iff it is applicable in the original sense of Definition 4, as described by the following theorem.

Theorem 10. *Let S be a tableau over \mathfrak{D} and $\mathcal{O} \in \mathcal{P}_{\text{fin}}(\mathfrak{D})$. For every set of labeled S -states \mathcal{M} , it holds that*

$$\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\text{ord}}} \mathcal{M} \quad \text{iff} \quad \{(\emptyset, \mathcal{O}_{\geq \ell})\} \xrightarrow{*}_S \mathcal{M},$$

where ℓ is the smallest label appearing in \mathcal{M} .

In particular, if S terminates in every input—as is the case for every tableau that is correct for some c-property—then its ordered extension must also terminate.

Corollary 11. *The ordered extension of any terminating tableau is terminating.*

More interesting, the execution of the ordered extension can be stopped as soon as the set of S -states \mathcal{M} is full of clashes, as further rule applications will not modify the clash degree.

Corollary 12. *Let $\mathcal{M}, \mathcal{M}'$ be sets of S -states such that \mathcal{M} is full of clashes and $\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\text{ord}}} \mathcal{M} \xrightarrow{*}_{S^{\text{ord}}} \mathcal{M}'$. Then $\ell_{\mathcal{M}} = \ell_{\mathcal{M}'}$.*

4.2 Dealing with Blocking

Consider now a tableau with blocking conditions associated to its rules, that is correct for a property \mathcal{P} . In general, the same blocking condition cannot be used for restricting rule applications in its labeled extension. This would lead to a clash degree that may be strictly smaller than the actual boundary that this extension tries to compute, as can be seen by a simple adaptation from the examples provided in [10] for subset blocking and in [4] for equality blocking. The reason for this is that, since the blocking condition is independent of the labels used, it might be that the assertions that trigger the blocking of a rule application may depend on different contexts.

To solve this problem, a modification of the blocking condition that also takes into account the labels was proposed in [10,4]. However, it is not clear whether more complex blocking conditions would require a more elaborate labeled extension. Moreover, we want to produce a boundary-computing algorithm that works for any kind of blocking condition that fits our very general framework.

Fortunately, if we consider the prioritized rule-application ordering of ordered extensions of tableaux, we can use Theorem 10 and Corollary 12 to show that the same blocking conditions yield a correct computation of the boundary.

To define the labeled extensions of tableaux with blocking, we adapt the notion of labeled applicability of a rule to take into account the blocking condition. An expansion rule R with a blocking condition \mathcal{B} is *labeled applicable* to a labeled S -state (A, \mathcal{O}) if it satisfies the three conditions from Definition 5 and additionally (iv) $A \notin \mathcal{B}$.

If a tableau with blocking is correct for a c-property \mathcal{P} , then it terminates on every input and when it reaches a saturated set of S -states, this is full of clashes iff the input ontology satisfied the property. We thus have the following result.

Theorem 13. *Let \mathcal{P} be a c-property on \mathfrak{D} and S a correct tableau (with blocking) for \mathcal{P} . For every ontology $\mathcal{O} \in \mathcal{P}_{\text{fin}}(\mathfrak{D})$ the following holds:*

- *the ordered extension of S terminates on \mathcal{O} ; that is, there is no infinite chain of rule applications $\{(\emptyset, \mathcal{O})\} \rightarrow_{S^{\text{ord}}} \mathcal{M}_1 \rightarrow_{S^{\text{ord}}} \mathcal{M}_2 \rightarrow_{S^{\text{ord}}} \dots$;*
- *for every chain of rule applications $\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\text{ord}}} \mathcal{M}$ with \mathcal{M} labeled saturated, the clash degree $\ell_{\mathcal{M}}$ induced by \mathcal{M} is a $(\mathcal{O}, \mathcal{P})$ -boundary.*

5 Conclusions

We have presented a general approach for extending general tableaux algorithms that can decide a c -property \mathcal{P} , into methods that can compute the boundary for \mathcal{P} w.r.t. to a set of linearly ordered contexts. Our approach can be used for extending the tableau-based algorithm for deciding consistency of \mathcal{ALC} ABoxes w.r.t. general TBoxes, but is not limited to this logic. In fact, adding transitive and inverse roles, or considering more complex blocking conditions would not be a problem for our framework. However, our notion of general tableaux requires a monotonic extension of the consequences deduced, and cannot identify previously introduced individuals in a clean way. In future work we will study whether we can construct a tableau for reasoning in full $SR\mathcal{OIQ}(\mathcal{D})$.

Our approach follows the lines of pinpointing extensions of tableaux, but has several advantages over these, mainly due to the simplicity of the linear ordering. By prioritizing the rule applications according to their degree—applying rules with higher degree first—we obtained an algorithm that does not differ much from the original tableau. The main intuition behind this prioritization of the rule applications is that it deduces all the consequences from a context higher in the ordering before testing contexts defined by a smaller label. Labeled rule applications using this ordering never modify the label of a previously existing assertion, and always add new assertions with non-increasing labels. Thus, chains of ordered rule applications correspond to rule applications of the original tableau. This in particular ensures that termination of a tableau transfers to its ordered extension, a property that is not shared by pinpointing extensions, or extensions based on lattices that are not linearly ordered.

The execution of ordered extensions of tableaux can be seen as incremental reasoning: given an ontology \mathcal{O} , the ordered extension first saturates the tableau using only the axioms having the greatest label ℓ_n . If this produces a set of S -states that is full of clashes, then we know that ℓ_n is the boundary for $(\mathcal{O}, \mathcal{P})$. Otherwise, it adds the axioms with label ℓ_{n-1} , and continues the tableau execution. Since several DL reasoners now implement incremental reasoning (e.g. Pellet,² FaCT++,³ or CEL⁴), we believe that an implementation of our glass-box approach for reasoning in DLs with linearly ordered contexts will be a simple step.

References

1. Baader, F., Buchheit, M., Hollunder, B.: Cardinality restrictions on concepts. *Artificial Intelligence* 88(1–2), 195–213 (1996)
2. Baader, F., Knechtel, M., Peñaloza, R.: Context-dependent views to axioms and consequences of semantic web ontologies. *Journal of Web Semantics* 12–13, 22–40 (April 2012), available at <http://dx.doi.org/10.1016/j.websem.2011.11.006>

² <http://clarkparsia.com/pellet/>

³ <http://code.google.com/p/factplusplus/>

⁴ <http://lat.inf.tu-dresden.de/systems/cel/>

3. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. *Journal of Automated Reasoning* 45(2), 91–129 (August 2010), special Issue: Selected Papers from IJCAR 2008
4. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. *Journal of Logic and Computation* 20(1), 5–34 (February 2010), special Issue: Tableaux and Analytic Proof Methods
5. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic \mathcal{EL}^+ . In: Hertzberg, J., Beetz, M., Englert, R. (eds.) *Proceedings of the 30th German Annual Conference on Artificial Intelligence (KI'07)*. *Lecture Notes in Artificial Intelligence*, vol. 4667, pp. 52–67. Springer-Verlag, Osnabrück, Germany (2007)
6. Davey, B.A., Priestley, H.A.: *Introduction to Lattices and Order*. Cambridge University Press, 2 edn. (2002)
7. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation* 9(3), 385–410 (1999)
8. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. *Journal of the Interest Group in Pure and Applied Logic* 8(3), 239–264 (2000)
9. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., Choi, K.S., Noy, N.F., Allemang, D., Lee, K.I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *Proc. of the 6th Int. Semantic Web Conf. and 2nd Asian Semantic Web Conf. (ISWC'07,ASWC'07)*. LNCS, vol. 4825, pp. 267–280. Springer-Verlag, Busan, Korea (2007)
10. Lee, K., Meyer, T., Pan, J.Z.: Computing maximally satisfiable terminologies for the description logic ALC with GCIs. In: Parsia, B., Sattler, U., Toman, D. (eds.) *Proc. of the 2006 Description Logic Workshop (DL'06)*. Lake District, UK (2006)
11. Lesot, M.J., Couchariere, O., Bouchon-Meunier, B., Rogier, J.L.: Inconsistency degree computation for possibilistic description logic: An extension of the tableau algorithm. In: *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS 2008)*. pp. 1–6. IEEE Computer Society Press (2008)
12. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Ellis, A., Hagino, T. (eds.) *Proc. of the 14th Int. Conf. on World Wide Web (WWW'05)*. pp. 633–640. ACM (2005)
13. Qi, G., Pan, J.Z.: A tableau algorithm for possibilistic description logic. In: Domingue, J., Anutariya, C. (eds.) *Proceedings of the 3rd Asian Semantic Web Conference (ASWC 2008)*. *Lecture Notes in Computer Science*, vol. 5367, pp. 61–75. Springer-Verlag (2008)
14. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*. pp. 355–362. Morgan Kaufmann, Los Altos, Acapulco, Mexico (2003)

Naïve ABox abduction in \mathcal{ALC} using a DL tableau

Ken Halland^{1,2} and Katarina Britz²

¹ School of Computing, University of South Africa, Pretoria, South Africa

² Centre for Artificial Intelligence Research: UKZN and CSIR Meraka Institute, South Africa

hallakj@unisa.ac.za and arina.britz@meraka.org.za

Abstract. The formal definition of abduction asks what needs to be added to a knowledge base to enable an observation to be entailed by the knowledge base. An observation which is not entailed by the knowledge base will result in open branches in a complete semantic tableau for the entailment. The statements required to close these branches therefore represent a solution to the abductive problem.

In this paper we describe how this idea can be implemented for ABox abduction in the description logic \mathcal{ALC} . We analyse the limitations of our algorithm and propose refinements to improve the quality of results.

1 Introduction

Abduction is a form of non-standard reasoning where explanations are generated for certain observations in the context of some background knowledge. This is as opposed to deduction – the standard form of reasoning where the logical consequences of some knowledge are determined.

A typical use of abduction is in the process of medical diagnosis. Say a patient displays some symptoms. A doctor uses abductive reasoning to generate hypotheses about the possible ailment(s) causing the symptoms. These hypotheses can then be tested by collecting corroborating evidence so that deduction can be used to make a correct diagnosis.

Abductive reasoning is *non-monotonic* in that the solutions derived from some background knowledge and an observation may no longer hold if we add new statements to the background knowledge.

Different forms of abduction have been formally defined in different logics. In their programmatic paper, Elsenbroich *et al* [6] define and describe various forms of abduction in description logics (DLs). A number of researchers have taken up the challenge and developed algorithms for some of these forms of abduction in selected DLs ([4, 5]).

In particular, Klarman *et al* [8] have provided a resolution-based algorithm and a tableau-based algorithm for performing ABox abduction in \mathcal{ALC} . The

tableau-based algorithm involves translating the knowledge base and the observation for an abductive problem from its DL specification to first-order logic (FOL), then constructing a FOL connection tableau and harvesting abductive solutions from open branches. These solutions are then translated back to a DL notation.

In this paper, we describe a glass box algorithm for doing this directly by means of a DL semantic tableau. We then show that this is a naïve algorithm in that it fails to generate a number of solutions which could be desirable in certain circumstances. We propose strategies to adapt the algorithm to address these deficiencies. We also define the notion of semantic minimality as a means to compare or rank solutions.

A potential advantage of this approach over Klarman’s is that it can utilise optimisation techniques used in DL tableaux.

Section 2 highlights the relevant aspects of the syntax and semantics of \mathcal{ALC} , Section 3 includes a definition of ABox abduction in \mathcal{ALC} (slightly more general than Klarman *et al* [8]), and Section 4 gives a description of the standard semantic tableau algorithm for \mathcal{ALC} . Section 5 then describes how this algorithm can be adapted to perform ABox abduction. Section 6 explains why the proposed abduction algorithm is naïve and how this can be addressed. Finally, Section 7 discusses the prospects for future work.

2 The Description Logic \mathcal{ALC}

Description Logics (DLs) are a family of fragments of first-order logic, suitable as knowledge representation formalisms and amenable to the implementation of efficient reasoners [1]. There is a trade-off between the expressivity of different DLs and the efficiency of the algorithms that have been defined to reason over them. \mathcal{ALC} is a DL of medium expressivity.

Syntax: The reader is referred to the Description Logic Handbook [1] for the syntax of \mathcal{ALC} . We highlight the following terminology for our current purposes:

A *knowledge base* is a set of *statements* partitioned into an *ABox* and a *TBox*. *ABox assertions* are statements of the form $C(I)$ and $R(I, J)$ (called *concept assertions* and *role assertions*, respectively), and *TBox axioms* are statements of the form $C \sqsubseteq D$ (sometimes called *general concept inclusions*, or *GCI*s).³

The following serves as a running example. Admittedly it is not very good knowledge representation, but we have chosen the given formulation for the sake of illustration.

Example 1: The following knowledge base is intended to express the ideas that Influenza A is a form of influenza, Malaria vivax is a form of malaria (caused

³ In this and the following specifications, C and D represent arbitrary concept descriptions, R represents an arbitrary role name, and I and J represent arbitrary individual names.

by *Plasmodium vivax*), and someone infected with influenza or malaria will be feverish:

Influenza(FLU_A)
Malaria(MAL_V)
 \exists infectedWith.Influenza \sqcup \exists infectedWith.Malaria \sqsubseteq Feverish

Semantics: Once again, the reader is referred to the DL Handbook [1] for the semantics of \mathcal{ALC} . We highlight the following terminology for our purposes:

An interpretation \mathcal{I} is a *model* of a knowledge base \mathcal{K} if all the statements of \mathcal{K} are true in \mathcal{I} . A concept description C is *satisfiable* with respect to a knowledge base \mathcal{K} if there is some model of \mathcal{K} such that the interpretation of C is not empty. A statement ϕ is *entailed* by a knowledge base \mathcal{K} if ϕ is true in all models of \mathcal{K} , in which case we write $\mathcal{K} \models \phi$. In an abuse of notation, we often write $\mathcal{K} \models \Phi$ where Φ is a set of statements. By this we mean that $\mathcal{K} \models \phi$ for all $\phi \in \Phi$. A knowledge base \mathcal{K} is *consistent* if it admits a model.

3 Abduction

An abduction problem is normally defined in terms of an observation (in the form of one or more statements) which is not entailed by a theory (i.e. a set of statements), and asks what needs to be added to the theory to entail the observation.

Example 2: Using the knowledge base given in Example 1, say we observe that John is feverish. An abduction problem would be to ask what should be added to the knowledge base to allow us to infer `Feverish(JOHN)`.

We expect abduction to allow us to hypothesize that John is infected with influenza or he is infected with malaria, i.e. \exists infectedWith.Influenza(JOHN) or \exists infectedWith.Malaria(JOHN). In fact, more specific hypotheses would be that he is infected with Influenza A, i.e. `infectedWith(JOHN,FLU_A)`, or that he is infected with Malaria vivax, i.e. `infectedWith(JOHN,MAL_V)`. The reader might like to check that adding any of these assertions to the knowledge base will allow us to infer `Feverish(JOHN)`.

One problem with a formal definition of abduction is how to narrow down the possibly infinite number of solutions to an abduction problem. Various criteria have been defined for this purpose. Like other authors [6, 8], we restrict our attention to the following three:

- i. *Consistency:* A solution should not create a contradiction with the background knowledge.
- ii. *Relevance:* A solution should be expressed in terms of the background knowledge; it shouldn't introduce an independent theory.
- iii. *Minimality:* A solution should not hypothesize more than necessary.

Example 3: The solutions given in Example 2 are consistent, relevant and minimal (i.e. minimal at least in a syntactic sense). The following solutions do not comply with these criteria:

- i. $\{\neg\text{Malaria}(\text{MAL}_V)\}$. If this assertion were added to the knowledge base, it would cause a contradiction, and would allow us to infer anything. But this would not be a helpful solution.
- ii. $\{\exists\text{infectedWith.ScarletFever} \sqsubseteq \text{Feverish}, \exists\text{infectedWith.ScarletFever}(\text{JOHN})\}$. This is an abductive solution, since if both these statements were added to the knowledge base, it would allow us to infer $\text{Feverish}(\text{JOHN})$. However, it would also allow us to make this inference *independently of the knowledge base* and is therefore not a relevant solution. (Incidentally, the observation itself, in this case $\{\text{Feverish}(\text{JOHN})\}$, is also always a non-relevant solution, since adding it to the knowledge base would allow the observation to be trivially inferred.)
- iii. $\{\exists\text{infectedWith.Influenza}(\text{JOHN}), \exists\text{infectedWith.Malaria}(\text{JOHN})\}$. Although this is a valid solution, it is not minimal because it hypothesizes too much, namely that John is infected with both influenza and malaria.

3.1 ABox Abduction in \mathcal{ALC}

As stated in the introduction, attempts have been made to define abduction and implement reasoners that can make abductive inferences in many logics, including description logics. ABox abduction (as opposed to general or so-called *knowledge base* abduction [6]) asks what ABox assertions need to be added to a DL knowledge base to allow an observation (also in the form of ABox assertions) to be inferred.

The astute reader will note that apart from not being relevant, Example 3 ii is also not an ABox abduction solution, since it contains a TBox axiom.

Definition 1. *Given a knowledge base \mathcal{K} and a set of ABox assertions Φ (both in \mathcal{ALC}) such that \mathcal{K} does not entail Φ and $\mathcal{K} \cup \Phi$ is consistent, then a set of ABox assertions Θ is an abductive solution for (\mathcal{K}, Φ) if $\mathcal{K} \cup \Theta \models \Phi$.*

We can narrow down the solutions in three ways:

- i. *Consistency:* $\mathcal{K} \cup \Theta$ is consistent.
- ii. *Relevance:* Φ is not entailed by Θ .
- iii. *Minimality:* We distinguish two kinds of minimality:
 - (a) *Syntactic:* No proper subset of Θ is a solution.
 - (b) *Semantic:* There is no non-equivalent solution Θ' such that $\mathcal{K} \cup \Theta \models \mathcal{K} \cup \Theta'$.

Note that our definition of semantic minimality induces a partial ordering on the set of solutions, and that there can be a number of semantically minimal (non-equivalent) solutions to a particular abductive problem. We say that a solution Θ is *closer to semantic minimality* than a solution Θ' if $\mathcal{K} \cup \Theta' \models \mathcal{K} \cup \Theta$ and $\mathcal{K} \cup \Theta \not\models \mathcal{K} \cup \Theta'$.

4 The Semantic Tableau Algorithm for \mathcal{ALC}

For a more detailed description of the semantic tableau algorithm for \mathcal{ALC} , the reader is referred to the Handbook of Knowledge Representation [2]. We highlight the following terminology for our current purposes:

The standard semantic tableau algorithm for description logics (and for \mathcal{ALC} in particular) tries to find (i.e. construct) a model of the knowledge base by applying so-called *expansion rules* to its statements.

The expansion rules only apply to ABox assertions, so before the algorithm can commence, the TBox axioms in the knowledge base must be converted to concept assertions by a process called *internalisation*.

In \mathcal{ALC} , it is possible to specify a knowledge base that has infinite models (by means of a so-called *cyclic* TBox). This issue is dealt with by a technique called *blocking*, which essentially detects when more than one individual has the same labelling in the current model.

If the algorithm detects a *contradiction* (or *clash*), i.e. the current set of assertions contains a concept assertion and its negation, it backtracks and tries another branch of its search. If it gets to a point where the current set of assertions are *saturated*, i.e. no more expansion rules can be applied and there is no contradiction, then a model has been found, the algorithm terminates and reports that the original knowledge base is consistent.

The algorithm described above performs *consistency checking* of a knowledge base. It can easily be adapted to perform the related reasoning task of *instance checking*, i.e. deciding whether a concept assertion is entailed by a knowledge base, as follows: The negation of the concept assertion being tested is added to the knowledge base and the algorithm described above is executed. If the resulting knowledge base is consistent, we conclude that the assertion is not entailed by the knowledge base (and *vice versa*).

5 Adaption of the Semantic Tableau Algorithm for ABox Abduction

For the purpose of ABox abduction, we perform instance checking of an observation by means of a so-called *extended* (or *complete*) semantic tableau, i.e. a tableau that doesn't terminate when the first open branch is attained. Every time an open branch is attained, the current set of assertions (representing a model of the original set of assertions) is stored, the algorithm backtracks and continues its search. Reiter's minimal hitting set algorithm [10] is then used to generate abductive solutions from these models. Simply put, one unexpanded concept assertion (involving a non-dummy individual) is chosen from each (terminal) open branch. (Dummy individuals are introduced by the \exists -expansion rule.) Each combination of the complements of such assertions forms an abductive solution. Finally the algorithm outputs all solutions that are consistent with the knowledge base and that are relevant.

<p>Input : ABox, TBox and Obs</p> <p>Output: Naïve abduction solutions</p> <pre> 1 A ← negNF(ABox) ∪ negNF(¬Obs); 2 internalise(TBox, U, A); 3 M ← {}; 4 SSet ← {}; 5 extendedST(A, U, M); 6 if M = {} then 7 print "The observation follows from the knowledge base"; 8 return SSet 9 minimalHS(M, H); 10 foreach S in H do 11 if consistentST(A ∪ S, U) and relevantST(S, Obs) then 12 SSet ← SSet ∪ {S} 13 return SSet </pre>
--

Algorithm 1: Naïve ABox abduction algorithm for \mathcal{ALC}

Function `negNF` transforms a set of concept assertions to negation normal form. Procedure `internalise` takes a set of TBox axioms and transforms them into a set of universal concepts U , each in negation normal form. (Note that in some implementations of the tableau algorithm, all the TBox axioms are converted into one long universal concept. We rather store them as separate concepts – one per TBox axiom – to save having to repeatedly expand the long concept.) The algorithm then applies each of these universal concepts to all the individual names mentioned in the ABox, adding the assertions to A . U is returned via parameter to be used in `extendedST` whenever a dummy variable is created for the \exists -rule. M is a set of models (where each model is a set of unexpanded assertions obtained from an open branch). Procedure `extendedST` performs the extended semantic tableau algorithm explained above. Whenever an open branch is attained, it adds the current set of unexpanded assertions to M and backtracks. If the observation is entailed by the knowledge base, then all branches will close and M will be empty. This means that we are not dealing with a proper abduction problem. M is sent to procedure `minimalHS` to generate the minimal hitting sets and store them in H . `minimalHS` ensures the syntactic minimality of solutions. Functions `consistentST` and `relevantST` are like calls to the semantic tableau procedure (described in Section 4). They determine whether the solution is consistent with the original knowledge base, and whether the observation is not entailed by the solution, respectively.

Although `extendedST` implements blocking, this is not used in any way for the generation of solutions. The argument is as follows: Since \mathcal{ALC} has the *finite model property* [1], every knowledge base that has an infinite model (handled by blocking) also has at least one finite model (represented by an open branch of the tableau). Since our algorithm closes all open branches and so removes all finite models, the infinite models will also be removed.

5.1 Complexity

The complexity of the standard semantic tableau algorithm for consistency checking with general TBoxes in \mathcal{ALC} is EXPTIME [2]. In our extended semantic tableau (in procedure `extendedST`), the worst case involves maximal branching where every branch is open, since we have to store all the assertion sets of all open branches. Nevertheless, the maximum number of branches is linear in the size of the initial assertion set, and the number of assertions in each such branch is also linear in the size of the initial assertion set. This means that the space required to store all the assertion sets in all the open branches is polynomial in the size of the initial assertion set. This at least means that the space requirements don't blow up to EXPSPACE, which means that the extended semantic tableau algorithm is at worst in EXPTIME.

Reiter's minimal hitting set algorithm (in general) is NP-complete [11]. In our case (in procedure `minimalHS`), the number of sets and their size is polynomial in the size of the initial assertion set. This means that the time required in our case is also in NP.

Finally, the algorithm invokes the functions `consistentST` and `relevantST` twice for each candidate solution. Although the space required for `relevantST` is only polynomial (because it does not deal with the TBox), `consistentST` is in EXPTIME in the worst case because it must deal with the TBox. Since the number of hitting sets is polynomial in the size of the initial assertion set, the total space requirement for this process is in EXPTIME.

The entire algorithm is therefore in EXPTIME.

5.2 Soundness and Completeness

Taking Definition 1 as the standard for ABox abduction in \mathcal{ALC} , Algorithm 1 is sound but not complete.

It is sound because all solutions that it generates are proper abduction solutions according to the definition. Consider the following argument: Each solution is a set comprised of the complements of assertions in the open branches of the extended semantic tableau, such that each open branch has a representative in the solution. So if the assertions of such a solution were to be added to the knowledge base (and the satisfiability test were to be performed again), all branches would close, indicating that the observation is now entailed by the knowledge base. This is precisely the definition of an abduction solution.

It is not complete due to the problems detailed in Section 6. One should not be surprised at this because abductive inference is notoriously incomplete due to the often infinite number of solutions to an abduction problem. Narrowing down the spectrum of solutions by means of criteria such as consistency, relevance and minimality only partially addresses this issue. Many solutions within these criteria are difficult to obtain, particularly by means of the techniques described here. In Section 6 we propose workarounds to make the algorithm more complete.

6 Naïvety

The algorithm described in Section 5 is “naïve” in that it doesn’t deal with all observations and it doesn’t generate all possible solutions allowed by our definition. In particular, the algorithm has the following weaknesses:

1. The observation can only consist of a single concept assertion.
2. Solutions containing disjunctions are not generated.
3. Solutions involving role assertions are not generated.
4. Semantically minimal solutions are not always generated.

We discuss each of these problems in turn:

6.1 Single Concept Assertions

Contrary to Definition 1, our algorithm does not allow more than one concept assertion in the observation. This is because an observation consisting of multiple assertions is really a conjunction of assertions, and the first step of the algorithm is to add the negation of the observation, which is in effect a disjunction of the negations of its individual assertions. We cannot express such a disjunction of assertions in DL syntax when the assertions involve different individuals. (This is not a problem for multiple concept assertions about the same individual, e.g. $C(I)$ and $D(I)$ can be negated as $\neg C \sqcup \neg D(I)$.)

Furthermore, the observation may not contain any role assertions, because \mathcal{ALC} syntax doesn’t allow negated role assertions. So we cannot deal with observations such as `hasSymptom(JOHN,INTERMITTENT_FEVER)`. (It is true that in many cases this situation could be handled by alternative modelling, e.g. $\exists \text{hasSymptom.} \text{IntermittentFever}(\text{JOHN})$, but there might be situations where this is not practical or desirable.)

Neither of these situations are a problem for Klarman *et al* [8], since their translation to FOL syntax allows disjunctions of concept assertions as well as negated role assertions.

Proposed Workaround: A brute-force method of dealing with multiple assertions in the observation would be to execute the algorithm once for each such assertion, harvest all the models from all open branches of all executions, and then process them as normal. This, however, would involve a lot of duplication (processing the rest of the assertions repeatedly). One way to avoid such duplication would be to keep the complemented assertions of the observation in a separate list from the other assertions. (The set of complemented assertions would represent a disjunction of assertions, whereas the other set would represent a conjunction of assertions – as normal.) Then when no other expansion rules can be applied to the normal set, the algorithm can branch for one of the complemented assertions.

The negation of a role assertion $R(I, J)$ can be accounted for by two assertions: $\forall R.A(I)$ and $\neg A(J)$, where A is a dummy concept name not occurring in

the knowledge base. Adding these two assertions to the initial set of assertions will have the same effect as adding the negation of the role assertion. Assertions involving such dummy concept names will need to be ignored for the purposes of determining abductive solutions.

6.2 Disjunctions

Our algorithm suffers from the same problem as Klarman’s [8], namely that the abductive solutions do not contain disjunctions, i.e. assertions of the form $C \sqcup D(I)$. For example, it does not generate the following solution to the problem described in Example 2: $(\exists \text{infectedWith.Influenza}) \sqcup (\exists \text{infectedWith.Malaria})(\text{JOHN})$. Note that a solution with such a disjunction is closer to semantic minimality than the corresponding two solutions with the individual disjuncts.

Proposed Workaround: One reason for this problem is that our algorithm only considers unexpanded concept assertions for forming solutions. Allowing expandable concept assertions to be selected for solutions would allow some disjunctions, but not all. For example, say we replaced the axiom of Example 1 with the two axioms $\exists \text{infectedWith.Influenza} \sqsubseteq \text{Feverish}$ and $\exists \text{infectedWith.Malaria} \sqsubseteq \text{Feverish}$. In this case, the solution with the disjunction above would be a valid solution, but would not be generated.

One could construct some such solutions from their constituent parts, e.g. $C \sqcup D(I)$ could be constructed from $C(I)$ and $D(I)$, but more complex solutions involving disjunctions inside quantifiers would be more difficult, e.g. $\exists R.(C \sqcup D)(I)$ will not be generated when $\exists R.C(I)$ and $\exists R.D(I)$ are.

Klarman *et al* [8] get around the problem by defining it away. They define ABox abduction in \mathcal{ALC} as only providing solutions in $\mathcal{AL}\mathcal{E}$ (a less expressive DL than \mathcal{ALC} , i.e. without disjunction and full negation).

6.3 Role Assertions

A more serious weakness is that the algorithm will never generate solutions involving role assertions. So the more specific solutions mentioned in Example 2, namely $\text{infectedWith}(\text{JOHN}, \text{FLU_A})$ and $\text{infectedWith}(\text{JOHN}, \text{MAL_V})$, are unattainable with our algorithm.

Stated more generically: Consider a knowledge base containing the ABox assertion $\forall R.A(I)$, and say we want an abductive explanation of the observation $A(J)$. An obvious solution is $R(I, J)$. But our abductive solutions are always the complements of assertions needed to close the open branches of a semantic tableau. Since the tableau algorithm does not infer negated role assertions, this solution will not be generated.

Proposed Workaround: One way would be to add the assertion $R(I, J)$ to a solution whenever the “pattern” $\{\forall R.A(I), \neg A(J)\}$ occurs in a open terminal branch. However, although this will enable some role assertions to be included in solutions, it will not generate all: Adding the role assertion $R(I, J)$ to a

knowledge base will cause an open branch containing $\{\forall R.C(I), \neg D(J)\}$, where C is disjoint from D , to close, so it should form part of an abductive solution whenever this pattern occurs. Such a pattern could be difficult to recognise, and the easiest way to deal with this would probably be to allow nominals in the language, since a negated role assertion $\neg R(I, J)$ can then be expressed as a concept assertion, namely $\forall R.\neg\{J\}(I)$ [7].

6.4 Semantic Minimality

This problem is best explained by means of an example. Say we add the axiom $\exists\text{bloodTestIndicates.Plasmodium} \sqsubseteq \exists\text{infectedWith.Malaria}$ to the knowledge base of Example 1.

Using the observation of Example 2, the algorithm now generates the solutions $\exists\text{infectedWith.Influenza}(\text{JOHN})$ and $\exists\text{bloodTestIndicates.Plasmodium}(\text{JOHN})$. One of the solutions we got previously, namely $\exists\text{infectedWith.Malaria}(\text{JOHN})$ has gone! In fact, a solution that is closer to semantic minimality has been lost.

Proposed Workaround: Many such solutions that are closer to semantic minimality can be obtained by allowing expanded concept assertions as part of solutions (including the above example). However, this will not solve all problems: Consider the knowledge base consisting of $\text{TBox} = \{A_1 \sqcup A_2 \sqsubseteq A_3, \exists R.A_3 \sqsubseteq A_4\}$ and $\text{ABox} = \{R(I, J)\}$, and say we want abductive solutions for the observation $\{A_4(I)\}$. If we apply the algorithm to this problem, three solutions are generated: $\{A_1(J)\}$, $\{A_2(J)\}$ and $\{A_3(J)\}$. If we allow expandable assertions, we get $\{A_1 \sqcup A_2(J)\}$ and $\{\exists R.A_3(I)\}$ as solutions, but not $\{\exists R.A_1(I)\}$ or $\{\exists R.A_2(I)\}$. These are closer to semantic minimality than $\{\exists R.A_3(I)\}$.

Whether we manage to find a way of generating all semantically minimal solutions, or just those attainable by allowing expandable assertions, we imagine that the user of a system implementing an abduction algorithm would want to be able to explore a range of such solutions.

The notion of semantic minimality is related to the notion of weakest sufficient conditions [9], although this work is restricted to propositional logic. It is also reminiscent of work on least common subsumers [3], and we plan to investigate the possibility of applying those ideas to this situation.

7 Future Work

Algorithm 1 does not implement many of the optimizations (e.g. back-jumping and caching) commonly used in DL tableau algorithms. Incorporating these into our algorithm promises to give a real efficiency advantage over the FOL connection tableau used in Klarman's algorithm [8].

This work also promises to be transferable to other more expressive DLs. As stated in Section 6.3, the problem of dealing with role assertions will disappear in languages that allow nominals.

Languages that do not have the finite model property will need some means of dealing with infinite models. (We imagine that the current assertion set at the point of blocking could simply be added to the set of models collected by `extendedST` so that it will be closed by all solutions.)

As stated in Section 6.4, we also intend to investigate the work on weakest sufficient conditions and least common subsumers for their applicability to ranking solutions.

Acknowledgements

This work was partially funded by a European Union international research staff exchange scheme – Project number 247601, Net2: Network for Enabling Networked Knowledge, from the FP7-PEOPLE-2009-IRSES call. Thanks to Tommie Meyer of the CSIR Meraka Institute (in South Africa) and Enrico Franconi of the Free University of Bozen/Bolzano (in Italy) for infrastructure and support.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook, Cambridge University Press (2003)
2. Baader, F., Horrocks, I. Sattler, U.: Chapter 3: Description Logics. In: van Harmelen, F., Lifschitz, V., Porter, B., editors: Handbook of Knowledge Representation, Elsevier (2007)
3. Baader, F., Sertkaya, B., Turhan, A.: Computing the least common subsumer w.r.t. a background terminology, Journal of Applied Logic, Springer (2004)
4. Di Noia, T., Di Sciascio, E., Donini, F.M.: Computing information minimal match explanations for logic-based matchmaking, in Proc. of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, vol 02, IEEE Computer Society (2009)
5. Du, J., Qi, G., Shen, Y-D., Pan, J.Z.: Towards practical ABox abduction in large OWL DL ontologies, in Proc. of the 25th AAAI Conference (2011)
6. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies, in Proc. of the OWLED'06 Workshop, vol 216 (2006)
7. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SR_{OLQ}*, in Proc. of KR2006, pp 57-67 (2006)
8. Klarman, S., Endriss, U., Schlobach, S.: ABox abduction in the description logic *ALC*, Journal of Automated Reasoning, vol 46:1 (2011)
9. Lin, F.: On strongest necessary and weakest sufficient conditions, in Proc. of KR2000, pp 167-175 (2000)
10. Reiter, R.: A theory of diagnosis from first principles, Artificial Intelligence, vol 32 (1987)
11. Wotawa, F.: A variant of Reiter's hitting-set algorithm, Information Processing Letters, vol 79 (2001)

Patent Valuation Using Difference in $\mathcal{AL}\mathcal{EN}$

Naouel Karam and Adrian Paschke

Department of Mathematics and Computer Science, Freie Universität Berlin
Königin-Luise-Str. 24-26, 14195 Berlin, Germany
naouel.karam@fu-berlin.de, paschke@inf.fu-berlin.de

Abstract. In this paper we present an approach for patent claim comparison based on the difference operator in description logics. The claims are represented using $\mathcal{AL}\mathcal{EN}$. An algorithm computing the difference between two $\mathcal{AL}\mathcal{EN}$ concept descriptions is proposed and its usefulness for patent application valuation is described by means of some simple examples.

1 Introduction

More and more, patent material is made available in electronic format, and most of the time in textual form. Examples of such services are the European patent Office (EPO) [1] and the United States Patent and Trademark Office (USPTO) [3]. The need for a formal specification of the patent content arose recently to help retrieval, examination and classification of patent applications.

In this context one needs to evaluate the innovative degree of a patent application. A claim is the part of a patent application where the inventor specifies the invention attributes and its features, defining what can be protected by the patent law. The aim is to show the non-obviousness to a person with basic domain skills and the novelty of the application compared to the state of the art.

Patent claims are described in natural language. Terms used in the claims must find clear support in the preceding description part of the patent application such that the meaning of a term is clearly specified. Previous work had attempted to represent patent material using OWL ontologies [7, 15].

Statistical tools performing patent application evaluation are based on a set of parameters provided by the user. Based on those parameters, the overall score of a patent application is calculated.

In this paper we present a more formal approach for patent claim comparison based on description logics¹. Patent application claims are formally represented as concept descriptions. When comparing two applications, the differences need to be pinpointed in order to prove the novelty. We compute this difference by using the difference operator and return the result to the user. The result must be intuitive and easy to understand in order to help the user construct his argumentation.

¹ This work has been partially supported by the Fact Screening and Transformation Project (FSTP) funded by the Teles Pri AG: www.fstp-expert-system.com

The difference operator computes the part of a concept that is not contained into another one. The difference operator has been defined in [14] as being the most general description that can be added to the second operand to obtain equivalence with the first. A second definition of the difference involving a syntactic criteria for minimality has been introduced in [4].

In our context we need to describe existential, universal and numerical restrictions. The difference operator has been investigated for description logics allowing either numerical restrictions [12] or existential restrictions [4]. We propose an algorithm for $\mathcal{AL}\mathcal{EN}$ based on the structural characterization of subsumption for $\mathcal{AL}\mathcal{EN}$ [10, 11].

The paper is organized as follows: in section 2, we give the formal definitions of the difference operator and discuss the motivation for our choice. Section 3 recalls useful results for $\mathcal{AL}\mathcal{EN}$ and introduces some notations. In section 4 we provide an algorithm for computing the difference in $\mathcal{AL}\mathcal{EN}$. Some examples of patent claims comparison are given in section 5 and Section 6 concludes the paper.

2 Operator Choice

The need for providing parts of a description that are not contained in another one has been motivated by different applications and different definitions exist with their advantages and drawbacks.

Informally speaking, the difference between two concept descriptions is the information contained in the first description and not in the second. The difference operator allows to remove from a given description all the information contained in another description. The difference operation between two concept descriptions was first introduced by Teege [14].

Definition 1. (*semantic difference*) *The difference between two concept descriptions C and D with $C \sqsubseteq D$ is given by*

$$C - D := \max\{E \mid E \sqcap D \equiv C\}$$

where \max is defined with respect to subsumption.

Later, the work in [9, 4] proposed a refinement of this definition by allowing the difference between incomparable descriptions (i.e. D is not required to subsume C) and taking the syntactic minimum (w.r.t. a subdescription ordering \preceq_d) instead of a semantic maximum.

Definition 2. (*syntactic difference*) *The difference between two incomparable concept descriptions C and D is defined as*

$$C - D := \min\{E \mid E \sqcap D \equiv C \sqcap D\}$$

where min is defined with respect to a so-called subdescription ordering (denoted by \preceq_d), used to compare syntactical structures that we recall in the next definition.

Definition 3. (*Subdescriptions*) Let C be an $\mathcal{AL}\mathcal{EN}$ -concept description, \widehat{C} is a subdescription of C ($\widehat{C} \preceq_d C$) iff,

1. $\widehat{C} = \perp$, or
2. \widehat{C} is obtained by removing from the top-level of C : \top , concept names, number restrictions, value restrictions, existential restrictions. For the remaining restrictions $\forall r.E$ or $\exists r.E$, replace the concept description E with a subdescription of E .

As an example, consider the $\mathcal{AL}\mathcal{EN}$ -concept description C :

$$P \sqcap P \sqcap \forall r.P \sqcap \exists r.(P \sqcap \exists r.Q) \sqcap \leq 3r$$

A possible subdescription:

$$\widehat{C} = P \sqcap \forall r.P \sqcap \exists r.(\exists r.Q) \sqcap \leq 3r$$

is obtained from C by removing P from the top-level of C and P from the subexpression $\exists r.(P \sqcap \exists r.Q)$. Note that \widehat{C} is equivalent to C and there exists no subdescription of \widehat{C} owning this property. \widehat{C} is the minimal subdescription which is equivalent to C .

In [12], the authors defined a semantic difference for $\mathcal{AL}\mathcal{N}$. They also compare the semantic and syntactic variants and state that the two approaches differ mainly on the bottom handling. Indeed, the bottom decomposition leads to more than one result for the semantic difference. On the other hand, the syntactic difference always generates a unique result that is a subdescription of the input description and hence more intuitive and comprehensive for the user.

Another operator called “concept abduction“ has been introduced to provide an explanation for matchmaking results in an e-marketplace. Abduction was first defined for $\mathcal{AL}\mathcal{N}$ [5] and has been recently extended to a more expressive DL \mathcal{SH} , using tableau calculus [13].

This operator does not require subsumption between the concept descriptions neither and thus is more general than the semantic difference. As stated in [6], the result of the difference is included in the set of solutions of a CAP.

In our context, we chose to use the syntactic difference for three reasons:

1. It does not require subsumption between the concept descriptions being compared, which is more realistic in real world applications,
2. it generates a unique result which is a syntactic variation of the input and hence more intuitive for the user,
3. when the inputs are free from unsatisfiable subexpressions the result is equivalent to the semantic difference.

3 Preliminaries

In this section we recall some results for subsumption in $\mathcal{AL}\mathcal{EN}$ and define some notations that will be useful to derive the difference algorithm.

The normal form of an $\mathcal{AL}\mathcal{EN}$ concept description is obtained by applying the following transformation rules:

- $(\geq mr) \sqcap (\geq nr) \rightarrow (\geq nr)$ if $n \geq m$,
- $(\leq mr) \sqcap (\leq nr) \rightarrow (\leq nr)$ if $n \leq m$, and
- $\forall r.C \sqcap \forall r.D \rightarrow \forall r.(C \sqcap D)$.

To access the different component of a concept description, we will use the following notations:

- $\text{prim}(C)$ denotes the set of concept names and the bottom concept occurring on the top-level of C ,
- $\text{inf}_r(C) = (\leq nr)$ if there exists a number restriction of the form $(\leq nr)$ on the top-level of C , $\text{inf}_r(C) = \top$ otherwise,
- $\text{sup}_r(C) = (\geq nr)$ if there exists a number restriction of the form $(\geq nr)$ on the top-level of C , $\text{sup}_r(C) = \top$ otherwise,
- $\text{min}_r(C) = \max\{k \mid C \sqsubseteq (\geq kr)\}$,
- $\text{max}_r(C) = \min\{k \mid C \sqsubseteq (\leq kr)\}$,
- $\forall_r(C) = E$ if there exists a value restriction $\forall r.E$ on the top-level of C ; $\forall_r(C) = \top$ otherwise,
- $\exists_r(C) = \{C' \mid \exists r.C' \text{ occurs on the top-level of } C\}$.

For the sake of clarity, we will assume that the set N_R is reduced to the role r . The algorithm can be easily generalized to arbitrary sets of role names.

The difference algorithm is based on the structural characterization of subsumption in $\mathcal{AL}\mathcal{EN}$ given in [10]. In the sequel we present the intuitions used to compute induced concept descriptions and recall the structural subsumption characterization. We also introduce some notations that we will use in the difference algorithm.

The main issue when dealing with $\mathcal{AL}\mathcal{EN}$ is to compute the "non-trivial" concept descriptions subsuming a concept description C ; such concepts do not appear in the top-level of C and are called "induced". Let us recall this process by means of examples.

Number restrictions:

They can be induced if two existential restrictions involve disjoint concepts. For example, if $C := \exists r.P \sqcap \exists r.\neg P$ then $\geq 2r$ is induced by C .

Let us note the induced number restrictions as follows: $\text{min}_r^*(C)$. Note that if $\text{min}_r^*(C) = k$, we have $C \sqsubseteq \geq kr$ which means that $\text{min}_r^*(C)$ is taken into account in $\text{min}_r(C)$ defined above.

Existential restrictions:

Due to \leq -number restrictions, if the number of existential restrictions in the top-level of a concept C is greater than the \leq -number restriction, those existential restrictions must be merged.

For example, if we have:

$$C := \exists r.(P \sqcap A) \sqcap \exists r.(P \sqcap B) \sqcap \exists r.(\neg P \sqcap A) \sqcap \exists r.Q \sqcap \exists r.\neg Q \sqcap \leq 2r,$$

the merging process results in new concepts descriptions where the only consistent ones are:

$$\begin{aligned} &\exists r.(P \sqcap Q \sqcap A \sqcap B) \sqcap \exists r.(\neg P \sqcap \neg Q \sqcap A) \leq 2r, \text{ and} \\ &\exists r.(P \sqcap \neg Q \sqcap A \sqcap B) \sqcap \exists r.(\neg P \sqcap Q \sqcap A) \leq 2r. \end{aligned}$$

In [10], each merging is represented by a mapping α leading to a concept C^α and the set of mappings is denoted by $\Gamma_r(C)$. The mappings in $\Gamma_r(C)$ are required to obey the following conditions:

1. $\alpha(i) \neq \emptyset$ for all $1 \leq i \leq n$;
2. $\bigcup_{1 \leq i \leq n} \alpha(i) = \{1, \dots, m\}$ and $\alpha(i) \cap \alpha(j) = \emptyset$ for all $1 \leq i < j \leq n$;
3. $\prod_{j \in \alpha(i)} C_j \sqcap \forall_r(C) \neq \perp$ for all $1 \leq i \leq n$,

where $n := \min\{\max_r(C), |\exists_r(C)|\}$ and $m := |\exists_r(C)|$.

In our example, this set consists of two mappings α_1 and α_2 leading to the two concepts C^{α_1} and C^{α_2} above with $C \equiv C^{\alpha_1} \sqcup C^{\alpha_2}$.

The set of C' such that $\exists r.C'$ in C^α for all mappings is noted as follows:

$$\exists_r^*(C) := \bigcup_{\alpha \in \Gamma_r(C)} \exists_r(C^\alpha)$$

For our example we obtain:

$$\exists_r^*(C) := \{P \sqcap Q \sqcap A \sqcap B, \neg P \sqcap \neg Q \sqcap A, P \sqcap \neg Q \sqcap A \sqcap B, \neg P \sqcap Q \sqcap A\}$$

In case $\exists_r(C) = \emptyset$ (there is no existential restriction on the top level of C) one needs to take into account the \geq -restrictions together with value restrictions to deduce non-trivial existential restrictions. Let us illustrate the case on the following concept:

$$D := (\geq 2r) \sqcap \forall r.(A \sqcap B).$$

The instances of D have at least two r -successors and due to the value restriction we can deduce that $D \sqsubseteq \exists r.(A \sqcap B)$.

This concludes the existential restrictions case.

Value restrictions:

It is clear from the former example that every instance of C has exactly two r -successors, in that case one can deduce from the existential restrictions in C^{α_1} and C^{α_2} that all r -successors belong to A and thus that the value restriction $\forall r.A$ is induced by C .

This deduction can be performed only when all r -successors are known and it is represented by the condition:

$$\max_r(C) = \min_r^*(C).$$

The second case where a value restriction can be induced is when $\max_r(C) = 0$ then $C \sqsubseteq \forall r.\perp$.

4 Syntactic Difference between $\mathcal{AL}\mathcal{EN}$ -Concept Descriptions

The algorithm computing the difference between two $\mathcal{AL}\mathcal{EN}$ -concept descriptions C and D is depicted in figure 1. We extended the algorithm proposed in [9] to compute the difference between two $\mathcal{AL}\mathcal{E}$ -concept descriptions.

Before going into detail about the algorithm, and as it is based on the structural characterization of the subsumption [10], let us recall the conditions for each kind of conjunct occurring on the top level of the concept descriptions.

Theorem 1. *Let C, D be two $\mathcal{AL}\mathcal{EN}$ -concept descriptions with $\exists_r(C) = \{C_1, \dots, C_n\}$. Then $C \sqsubseteq D$ iff $C \equiv \perp$, or $D \equiv \top$, or the following conditions hold:*

1. $\text{prim}(C) \subseteq \text{prim}(D)$,
2. $\text{max}_r(C) \leq \text{max}_r(D)$,
3. $\text{min}_r(C) \geq \text{min}_r(D)$,
4. for all $D' \in \exists_r(D)$ it holds that
 - (a) $\exists_r(C) = \emptyset$, $\text{min}_r(C) \geq 1$, and $\forall_r(C) \sqsubseteq D'$; or
 - (b) $\exists_r(C) \neq \emptyset$, and for each $\alpha \in \Gamma_r(C)$, there exists $C' \in \exists_r(C^\alpha)$ such that $C' \sqcap \forall_r(C) \sqsubseteq D'$.
5. if $\forall_r(C) \neq \top$, then
 - (a) $\text{max}_r(C) = 0$; or
 - (b) $\text{min}_r(C) < \text{max}_r(C)$ and $\forall_r(C) \sqsubseteq \forall_r(D)$; or
 - (c) $0 < \text{min}_r(C) = \text{max}_r(C)$ and $\forall_r(C) \sqcap C' \sqsubseteq \forall_r(D)$ for all $C' \in \exists_r^*(C)$.

According to the definition of the difference (c.f Definition 2), the output of the algorithm must verify:

$$\text{ComputeDiff}(C, D) \sqcap D \equiv C \sqcap D$$

Let start with concept names and numerical restrictions. One can easily verify conditions 1.–3. of Theorem 1:

- The set of primitive concepts $\text{prim}(\text{ComputeDiff}(C, D) \sqcap D)$ is equal to $\text{prim}(\text{ComputeDiff}(C, D)) \cup \text{prim}(D)$, which by definition is equal to $(\text{prim}(C) \setminus \text{prim}(D)) \cup \text{prim}(D)$. This is again equal to $\text{prim}(C) \cup \text{prim}(D)$, the set of primitive concepts of $(C \sqcap D)$.
- For $\text{max}_r(\text{ComputeDiff}(C, D) \sqcap D)$, we distinguish two cases: it is equal to $\text{max}_r(D)$, when $\text{max}_r(C) \geq \text{max}_r(D)$ and to $\text{max}_r(\text{inf}_r(C) \sqcap D)$, when $\text{max}_r(C) < \text{max}_r(D)$. Both cases are equal to $\text{max}_r(C \sqcap D)$.
- Analogously to the precedent case, for $\text{min}_r(\text{ComputeDiff}(C, D) \sqcap D)$ we have two cases: equal to $\text{min}_r(D)$ when $\text{min}_r(C) \leq \text{min}_r(D)$ and to $\text{min}_r(\text{sup}_r(C) \sqcap D)$ when $\text{min}_r(C) > \text{min}_r(D)$. Which again are equal to $\text{min}_r(C \sqcap D)$.

Require: Two $\mathcal{AL}\mathcal{EN}$ -concept descriptions C and D in $\mathcal{AL}\mathcal{EN}$ -normal form
Ensure: $\text{ComputeDiff}(C, D)$

- 1: **if** $C \sqcap D \equiv \perp$ **then**
- 2: $\text{ComputeDiff}(C, D) := \perp$
- 3: **else**
- 4: $\text{ComputeDiff}(C, D) := \bigcap_{A \in \text{prim}(C-D)} A \sqcap \text{inf}_r(C-D) \sqcap \text{sup}_r(C-D) \sqcap \forall r. \text{ComputeDiff}(\forall_r(C), \forall_r(D) \sqcap \forall_r^*(C \sqcap D)) \sqcap \bigcap_{E \in \mathcal{E}'_r} \exists r.E$,
 where the value restriction is omitted in case $\text{ComputeDiff}(\forall_r(C), \forall_r(D) \sqcap \forall_r^*(C \sqcap D)) \equiv \top$ and:
 - $\text{prim}(C-D) := \text{prim}(C) \setminus \text{prim}(D)$;
 - $\text{inf}_r(C-D) = \begin{cases} \top, & \max_r(C) \geq \max_r(D), \\ \text{inf}_r(C), & \max_r(C) < \max_r(D); \end{cases}$
 - $\text{sup}_r(C-D) = \begin{cases} \top, & \min_r(C) \leq \min_r(D), \\ \text{sup}_r(C), & \min_r(C) > \min_r(D); \end{cases}$
 - $\forall_r^*(E) = \begin{cases} \top, & \min_r^*(E) < \max_r(E), \\ \perp, & \max_r(E) = 0, \\ \text{lcs}(\{\forall_r(E) \sqcap E' \mid E' \in \exists_r^*(E)\}), & 0 < \min_r^*(E) = \max_r(E); \end{cases}$
 - \mathcal{E}'_r is computed as follows:

Let $\mathcal{E}_r := \exists_r(C) = \{C_1, \dots, C_n\}$ and we define $C^{\setminus E}$ as being C without the conjunct $\exists r.E$.
- 5: **for** $i = 1$ to n **do**
- 6: **if** (i) $\Gamma_r(C^{\setminus C_i} \sqcap D) \neq \emptyset$ and there exists $D' \in \exists_r((C^{\setminus C_i} \sqcap D)^\alpha)$ for all $\alpha \in \Gamma_r(C^{\setminus C_i} \sqcap D)$ with $\forall_r(C) \sqcap \forall_r(D) \sqcap D' \sqsubseteq C_i$, or
 (ii) $\Gamma_r(C^{\setminus C_i} \sqcap D) = \emptyset$ and $\min_r(C \sqcap D) \geq 1$ and $\forall_r(C) \sqcap \forall_r(D) \sqsubseteq C_i$,
 then
- 7: $\mathcal{E}_r := \mathcal{E}_r \setminus \{C_i\}$;
- 8: **end if**
- 9: **end for**
- 10: $\mathcal{E}'_r = \{E^* \mid E \in \mathcal{E}_r\}$ where $E^* := \text{ComputeDiff}(E, \forall_r(C) \sqcap \forall_r(D))$.
- 11: **end if**

Fig. 1. The algorithm ComputeDiff

Let us now consider the value restrictions. In case $0 < \min_r^*(C \sqcap D) = \max_r(C \sqcap D)$, a value restriction $\forall_r(C \sqcap D)$ can be deduced from $C \sqcap D$, which is the least common subsumer of all the existential restrictions appearing in $\exists_r^*(C \sqcap D)$.

Let us illustrate this case by an example.

$$C_{ex1} := \exists r.(P \sqcap A) \sqcap \exists r.(P \sqcap B) \sqcap \exists r.\neg Q \sqcap \forall r.(A \sqcap B),$$

$$D_{ex1} := \exists r.(\neg P \sqcap A) \sqcap \exists r.Q \sqcap (\leq 2r).$$

As demonstrated in section 3, the merging of existential restrictions in $C_{ex1} \sqcap D_{ex1}$ induces the value restriction $\forall_r.A$. The value restriction of the difference $C-D$ is then

$$\text{ComputeDiff}(\forall_r(C), \forall_r(D) \sqcap \forall_r^*(C \sqcap D)) = \text{ComputeDiff}(A \sqcap B, A) = B$$

One can verify condition 5(c) of Theorem 1, namely that $B \sqcap C' \sqsubseteq A \sqcap B$ for all C' in the set of merged existential restrictions $\exists_r^*(C_{ex1} \sqcap D_{ex1})$ (c.f. the example in Section 3).

Let us now illustrate the two cases for existential restrictions by means of examples.

- To illustrate case (i) let us take the following descriptions:

$$\begin{aligned} C_{ex2} &:= \exists r.(P \sqcap A \sqcap B) \sqcap \leq 2r, \\ D_{ex2} &:= \exists r.(P \sqcap A) \sqcap \exists r.(P \sqcap B) \sqcap \exists r.(\neg P \sqcap A) \sqcap \exists r.Q \sqcap \exists r.\neg Q. \end{aligned}$$

$C_1 = P \sqcap A \sqcap B$ subsumes $P \sqcap Q \sqcap A \sqcap B$ from $(C_{ex2}^{\setminus C_1} \sqcap D_{ex2})^{\alpha 1}$ and $P \sqcap \neg Q \sqcap A \sqcap B$ from $(C_{ex2}^{\setminus C_1} \sqcap D_{ex2})^{\alpha 2}$. We then have

$$\text{ComputeDiff}(C_{ex2}, D_{ex2}) = \leq 2r.$$

- Finally we illustrate case (ii) with the following example

$$\begin{aligned} C_{ex3} &:= P \sqcap \exists r.(A_1 \sqcap A_2) \sqcap (\geq 3r), \\ D_{ex3} &:= \forall r.(A_1 \sqcap A_2 \sqcap A_3). \end{aligned}$$

$$\text{ComputeDiff}(C_{ex3}, D_{ex3}) = P \sqcap (\geq 3r)$$

While $\exists r.(A_1 \sqcap A_2 \sqcap A_3)$ can be induced from $\forall r.(A_1 \sqcap A_2 \sqcap A_3) \sqcap (\geq 3r)$, one can verify that

$$\begin{aligned} \text{ComputeDiff}(C_{ex3}, D_{ex3}) \sqcap D_{ex3} &\equiv P \sqcap (\geq 3r) \sqcap \forall r.(A_1 \sqcap A_2 \sqcap A_3) \\ &\sqsubseteq P \sqcap (\geq 3r) \sqcap \exists r.(A_1 \sqcap A_2 \sqcap A_3) \sqcap \forall r.(A_1 \sqcap A_2 \sqcap A_3) \\ &\sqsubseteq P \sqcap (\geq 3r) \sqcap \exists r.(A_1 \sqcap A_2) \sqcap \forall r.(A_1 \sqcap A_2 \sqcap A_3) \\ &\equiv C_{ex3} \sqcap D_{ex3} \end{aligned}$$

The case \sqsupseteq is obvious.

The next lemma proves that the algorithm returns a difference that respects the condition of the difference operator (Section 2). The proof can be found in our technical report [8].

Lemma 1. *Let C, D be two $\mathcal{AL}\mathcal{EN}$ -concept descriptions in $\mathcal{AL}\mathcal{EN}$ normal form. Then, $\text{ComputeDiff}(C, D) \sqcap D \equiv C \sqcap D$.*

5 Application to Patent Claims Comparison

In this section we are going to illustrate the difference on some simple patent examples.

Example 1. Suppose we have an application for a chair with only one leg having a seat made only from a light material. The corresponding concept description is the following:

$$= 1hasLeg \sqcap \exists hasSeat. (\forall hasMaterial. Light),$$

that we need to compare to a previous patent describing a chair with three legs and having one seat made of light wood, given by the description

$$= 3hasLeg \sqcap \exists hasSeat. (\forall hasMaterial. (Wood \sqcap Light)).$$

The algorithm returns $\leq 1hasLeg$ which corresponds to the specific part of the application.

Example 2. Let us consider a patent application for a watch with two types of displays or more that are all bright. The corresponding description is:

$$\geq 2hasDisplay \sqcap \forall hasDisplay. Bright,$$

to compare to a watch with an analogue and a non analogue display described as:

$$\exists hasDisplay. Analogical \sqcap \exists hasDisplay. \neg Analogical.$$

The difference is $\forall hasDisplay. Bright$. Indeed, we can deduce the numerical restriction $\geq 2hasDisplay$ from the second concept description while the existential restrictions involve disjoint concepts.

6 Conclusion

In this paper, we have investigated the problem of computing the syntactic difference in $\mathcal{AL}\mathcal{EN}$. This work was motivated by an application in the context of patent applications valuation. In this context one needs to compare the claims of the patent with previous patents solving a similar problem. The textual descriptions of the claims can be described in a formal way using $\mathcal{AL}\mathcal{EN}$ -concept descriptions. The difference operator provides the user with the parts that are contained in his application and not in the state of the art. Our aim is not to provide a decision-making tool regarding the novelty of a patent application, but rather a tool that can help in pinpointing the differences and assist the user in constructing his argumentation.

We are implementing a prototype of the difference algorithm based on the DL reasoner HerMiT [2] for the subsumption test. A direction for Future work would be to investigate the decision making process based on the results returned by the difference and empirical rules derived from experts decisions.

References

1. The European Patent Office website: <http://www.epo.org>
2. The Hermit OWL Reasoner website: <http://hermit-reasoner.com>
3. The United States Patent and Trademark Office website: <http://www.uspto.gov/>
4. Brandt, S., Kusters, R., Turhan, A.Y.: Approximation and difference in description logics. In: T.Dean (ed.) Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002). pp. 203–214. Morgan Kaufmann, San Francisco, CA (2002)
5. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Concept abduction and contraction in description logics. In: Description Logics (2003)
6. Di Noia, T., Di Sciascio, E., Donini, F.M.: Semantic matchmaking as non-monotonic reasoning: a description logic approach. *J. Artif. Int. Res.* 29(1), 269–307 (2007), <http://dl.acm.org/citation.cfm?id=1622606.1622615>
7. Giereth, M., Koch, S., Kompatsiaris, Y., Papadopoulos, S., Pianta, E., Serafini, L., Wanner, L.: A modular framework for ontology-based representation of patent information. In: JURIX (2007)
8. Karam, N., Paschke, A.: Patent valuation using difference in ALEN. Tech. rep., Freie Universitat Berlin, Germany (2012), <http://www.csw.inf.fu-berlin.de/publications/TR-DiffAlen.pdf>
9. Kusters, R.: Non-standard inferences in description logics, vol. 2100 of Lecture Notes in Artificial Intelligence. Springer-Verlag (2001)
10. Kusters, R., Molitor, R.: Computing least common subsumers in ALEN. In: Proceedings of the 17th international joint conference on Artificial intelligence - Volume 1. pp. 219–224. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001), <http://portal.acm.org/citation.cfm?id=1642090.1642120>
11. Kusters, R., Molitor, R.: Structural subsumption and least common subsumers in a description logic with existential and number restrictions. *Studia Logica* 81(2), 227–259 (2005)
12. Leger, A., Rey, C., Toumani, F.: Semantic difference in ALN. In: Description Logics (2007)
13. Noia, T.D., Sciascio, E.D., Donini, F.M.: A tableaux-based calculus for abduction in expressive description logics: preliminary results. In: Description Logics (2009)
14. Teege, G.: Making the difference: a subtraction operation for description logics. In: Doyle, J., Sandewall, E., Torasso, P. (eds.) KR'94: Principles of Knowledge Representation and Reasoning, pp. 540–550. Morgan Kaufmann, San Francisco, California (1994)
15. Wanner, L., Baeza-Yates, R., Brüggemann, S., Codina, J., Diallo, B., Escorsa, E., Giereth, M., Kompatsiaris, Y., Papadopoulos, S., Pianta, E., Piella, G., Puhlmann, I., Rao, G., Rotard, M., Schoester, P., Serafini, L., Zervaki, V.: Towards content-oriented patent document processing. *World Patent Information* 30(1), 21–33 (2008)

A Formal Characterization of Concept Learning in Description Logics

Francesca A. Lisi

Dipartimento di Informatica, Università degli Studi di Bari “Aldo Moro”, Italy
lisi@di.uniba.it

Abstract. Among the inferences studied in Description Logics (DLs), induction has been paid increasing attention over the last decade. Indeed, useful non-standard reasoning tasks can be based on the inductive inference. Among them, **Concept Learning** is about the automated induction of a description for a given concept starting from classified instances of the concept. In this paper we present a formal characterization of **Concept Learning** in DLs which relies on recent results in Knowledge Representation and Machine Learning.

1 Introduction

Building and maintaining large ontologies pose several challenges to Knowledge Representation (KR) because of their size. In DL ontologies, although standard inferences help structuring the knowledge base (KB), e.g., by automatically building a concept hierarchy, they are, for example, not sufficient when it comes to (automatically) generating new concept descriptions from given ones. They also fail if the concepts are specified using different vocabularies (i.e. sets of concept names and role names) or if they are described on different levels of abstraction. Altogether it has turned out that for building and maintaining large DL KBs, besides the standard inferences, additional so-called *non-standard inferences* are required [27,19]. Among them, the first ones to be studied have been the Least Common Subsumer (LCS) of a set concepts [3] and the Most Specific Concept (MSC) of an individual [32,10,20,1]. Very recently, a unified framework for non-standard reasoning services in DLs has been proposed [8]. It is based on the use of second-order sentences in DLs [7] as the unifying definition model for all those *constructive reasoning* tasks which rely on specific optimality criteria to build up the objective concept. E.g., LCS is one of the cases considered for one such reformulation in terms of optimal solution problems.

Since [27], much work has been done in DL reasoning to support the construction and maintenance of DL KBs. This work has been more or less explicitly related to *induction*. E.g., the notion of LCS has subsequently been used for the bottom-up induction of CLASSIC concept descriptions from examples [5,6]. Induction has been widely studied in Machine Learning (ML). Therefore it does not come as a surprise that the problem of finding an appropriate concept description for given concept instances, reformulated as a problem of inductive

learning from examples, has been faced in ML, initially attacked by heuristic means [6,18,14] and more recently in a formal manner [2,12,13,22] by adopting the methods and the techniques of that ML approach known as **Concept Learning**.

In this paper, we present a formal characterization of **Concept Learning** in DLs which relies on recent results in KR and ML. Notably, the proposed formulation can be justified by observing that the inductive inference deals with finding - or constructing - a concept. Therefore, non-standard reasoning services based on induction can be considered as constructive reasoning tasks. Starting from this assumption, and inspired by Colucci *et al*'s framework, **Concept Learning** is modeled as a second-order concept expression in DLs and reformulated in terms that allow for a construction possibly subject to some optimality criteria.

The paper is structured as follows. Section 2 is devoted to preliminaries on **Concept Learning** according to the ML tradition. Section 3 defines the **Concept Learning** problem statement in the KR context. Section 4 proposes a reformulation of **Concept Learning** as a constructive reasoning task in DLs. Section 5 concludes the paper with final remarks and directions of future work.

2 Preliminaries

2.1 Machine Learning

The goal of ML is the design and development of algorithms that allow computers to evolve behaviors based on empirical data [30]. The automation of the *inductive inference* plays a key role in ML algorithms, though other inferences such as abduction and analogy are also considered. The effect of applying inductive ML algorithms depends on whether the *scope* of induction is discrimination or characterization [28]. *Discriminant induction* aims at inducing hypotheses with discriminant power as required in tasks such as classification. In classification, observations to learn from are labeled as positive or negative instances of a given class. *Characteristic induction* is more suitable for finding regularities in a data set. This corresponds to learning from positive examples only.

Ideally, the ML task is to discover an operational description of a *target function* $f : X \rightarrow Y$ which maps elements in the *instance space* X to the values of a set Y . The target function is unknown, meaning that only a set \mathcal{D} (the *training data*) of points of the form $(x, f(x))$ is provided. However, it may be very difficult in general to learn such a description of f perfectly. In fact, ML algorithms are often expected to acquire only some approximation \hat{f} to f by searching a very large space \mathcal{H} of possible hypotheses (the *hypothesis space*) which depend on the representation chosen for f (the *language of hypotheses*). The output approximation is the one that best fits \mathcal{D} according to a *scoring function* $score(f, \mathcal{D})$. It is assumed that any hypothesis $h \in \mathcal{H}$ that approximates f well w.r.t. a large set of training cases will also approximate it well for new unobserved cases. These notions have been mathematically formalized in computational learning theory within the Probably Approximately Correct (PAC) learning framework [36].

Summing up, given a hypothesis space \mathcal{H} and a training data set \mathcal{D} , ML algorithms are designed to find an approximation \hat{f} of a target function f s.t.:

1. $\hat{f} \in \mathcal{H}$;
2. $\hat{f}(\mathcal{D}) \approx f(\mathcal{D})$; and/or
3. $\hat{f} = \operatorname{argmax}_{f \in \mathcal{H}} \operatorname{score}(f, \mathcal{D})$.

It has been recently stressed that the first two requirements impose constraints on the possible hypotheses, thus defining a *Constraint Satisfaction Problem* (CSP), whereas the third requirement involves the optimization step, thus turning the CSP into an *Optimization Problem* (OP) [9]. We shall refer to the ensemble of constraints and optimization criterion as the model of the learning task. Models are almost by definition declarative and it is useful to distinguish the CSP, which is concerned with finding a solution that satisfies all the constraints in the model, from the OP, where one also must guarantee that the found solution be optimal w.r.t. the optimization function. Examples of typical CSPs in the ML context include **Concept Learning** for reasons that will become clearer by reading the following subsection.

2.2 Concept Learning

Concept Learning deals with inferring the general definition of a category based on members (positive examples) and nonmembers (negative examples) of this category. Here, the target is a boolean-valued function $f : X \rightarrow \{0, 1\}$, *i.e.* a *concept*. When examples of the target concept are available, the resulting ML task is said *supervised*, otherwise it is called *unsupervised*. The positive examples are those instances with $f(x) = 1$, and negative ones are those with $f(x) = 0$.

In **Concept Learning**, the key inferential mechanism for induction is *generalization as search* through a partially ordered space of inductive hypotheses [29]. Hypotheses may be ordered from the most general ones to the most specific ones. We say that an instance $x \in X$ satisfies a hypothesis $h \in \mathcal{H}$ if and only if $h(x) = 1$. Given two hypotheses h_i and h_j , h_i is more general than or equal to h_j (written $h_i \succeq_g h_j$, where \succeq_g denotes a *generality relation*) if and only if any instance satisfying h_j , also satisfies h_i . Note that it may not be always possible to compare two hypotheses with a generality relation: the instances satisfied by the hypotheses may intersect, and not necessarily be subsumed by one another. The relation \succeq_g defines a *partial order* (*i.e.*, it is reflexive, antisymmetric, and transitive) over the space of hypotheses.

A hypothesis h that correctly classifies all training examples is called consistent with these examples. For a consistent hypothesis h it holds that $h(x) = f(x)$ for each instance x . The set of all hypotheses consistent with the training examples is called the *version space* with respect to \mathcal{H} and \mathcal{D} . **Concept Learning** algorithms may use the hypothesis space structure to efficiently search for relevant hypotheses. E.g., they may perform a specific-to-general search through the hypothesis space along one branch of the partial ordering, to find the most specific hypothesis consistent with the training examples. Another well known approach, *candidate elimination*, consists of computing the version space by an incremental computation of the sets of maximally specific and maximally general hypotheses. An important issue in **Concept Learning** is associated with the

so-called *inductive bias*, *i.e.* the set of assumptions that the learning algorithm uses for prediction of outputs given previously unseen inputs. These assumptions represent the nature of the target function, so the learning approach implicitly makes assumptions on the correct output for unseen examples.

Inductive Logic Programming (ILP) was born at the intersection between **Concept Learning** and the field of Logic Programming [31]. From **Concept Learning** it has inherited the inferential mechanisms for induction [33]. However, a distinguishing feature of ILP with respect to other forms of **Concept Learning** is the use of prior knowledge of the domain of interest, called *background knowledge* (BK), during the search for hypotheses. Due to the roots in Logic Programming, ILP was originally concerned with **Concept Learning** problems where both hypotheses, observations and BK are expressed with first-order Horn rules (usually DATALOG for computational reasons). E.g., FOIL is a popular ILP algorithm for learning sets of DATALOG rules for classification purposes [34]. It performs a greedy search in order to maximize an *information gain* function. Therefore, FOIL implements an OP version of **Concept Learning**.

Over the last decade, ILP has widened its scope significantly, by considering, e.g., learning in DLs (see next section) as well as within those hybrid KR frameworks integrating DLs and first-order clausal languages [35,17,25,26].

3 Learning Concepts in Description Logics

Early work on the application of ML to DLs essentially focused on demonstrating the PAC-learnability for various terminological languages derived from CLASSIC. In particular, Cohen and Hirsh investigate the CORECLASSIC DL proving that it is not PAC-learnable [4] as well as demonstrating the PAC-learnability of its sub-languages, such as C-CLASSIC [5], through the bottom-up LCSLEARN algorithm. It is also worth mentioning unsupervised learning methodologies for DL concept descriptions, whose prototypical example is KLUSTER [18], a polynomial-time algorithm for the induction of BACK terminologies. More recently, algorithms have been proposed that follow the *generalization as search* approach by extending the methodological apparatus of ILP to DL languages [2,11,12,21,22]. Supervised (resp., unsupervised) learning systems, such as YINYANG [16] and \mathcal{DL} -LEARNER [23], have been implemented. Based on a set of refinement operators borrowed from YINYANG and \mathcal{DL} -LEARNER, a new version of the FOIL algorithm, named \mathcal{DL} -FOIL, has been proposed [13]. In \mathcal{DL} -FOIL, the information gain function takes into account the Open World Assumption (OWA) holding in DLs. Indeed, many instances may be available which cannot be ascribed to the target concept nor to its negation. This requires a different setting to ensure a special treatment of the unlabeled individuals.

3.1 The Problem Statement

In this section, the supervised **Concept Learning** problem in the DL setting is formally defined. For the purpose, we denote:

- \mathcal{T} and \mathcal{A} are the TBox and the ABox, respectively, of a \mathcal{DL} KB \mathcal{K}
- $\text{Ind}(\mathcal{A})$ is the set of all individuals occurring in \mathcal{A}
- $\text{Retr}_{\mathcal{K}}(C)$ is the set of all individuals occurring in \mathcal{A} that are an instance of a given concept C w.r.t. \mathcal{T}
- $\text{Ind}_C^+(\mathcal{A}) = \{a \in \text{Ind}(\mathcal{A}) \mid C(a) \in \mathcal{A}\} \subseteq \text{Retr}_{\mathcal{K}}(C)$
- $\text{Ind}_C^-(\mathcal{A}) = \{b \in \text{Ind}(\mathcal{A}) \mid \neg C(b) \in \mathcal{A}\} \subseteq \text{Retr}_{\mathcal{K}}(\neg C)$

These sets can be easily computed by resorting to *retrieval inference services* usually available in DL systems.

Definition 1 (Concept Learning). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a \mathcal{DL} KB. Given:

- a (new) target concept name C
- a set of positive and negative examples $\text{Ind}_C^+(\mathcal{A}) \cup \text{Ind}_C^-(\mathcal{A}) \subseteq \text{Ind}(\mathcal{A})$ for C
- a concept description language $\mathcal{DL}_{\mathcal{H}}$

the **Concept Learning problem** is to find a concept definition $C \equiv D$ such that $D \in \mathcal{DL}_{\mathcal{H}}$ satisfies the following conditions

- Completeness** $\mathcal{K} \models D(a) \quad \forall a \in \text{Ind}_C^+(\mathcal{A})$ and
Consistency $\mathcal{K} \models \neg D(b) \quad \forall b \in \text{Ind}_C^-(\mathcal{A})$

Note that the definition given above provides the CSP version of the supervised **Concept Learning** problem. However, as already mentioned, **Concept Learning** can be regarded also as an OP. Algorithms such as \mathcal{DL} -FOIL testify the existence of optimality criteria to be fulfilled in **Concept Learning** besides the conditions of completeness and consistency.

3.2 The Solution Strategy

In Def. 1, we have considered a language of hypotheses $\mathcal{DL}_{\mathcal{H}}$ that allows for the generation of concept definitions in any \mathcal{DL} . These definitions can be organized according to the concept subsumption relationship \sqsubseteq . Since \sqsubseteq induces a quasi-order (*i.e.*, a reflexive and transitive relation) on $\mathcal{DL}_{\mathcal{H}}$ [2,11], the problem stated in Def. 1 can be cast as the search for a correct (*i.e.*, complete and consistent) concept definition in $(\mathcal{DL}_{\mathcal{H}}, \sqsubseteq)$ according to the *generalization as search* approach in Mitchell’s vision. In such a setting, one can define suitable techniques (called *refinement operators*) to traverse $(\mathcal{DL}_{\mathcal{H}}, \sqsubseteq)$ either top-down or bottom-up.

Definition 2 (Refinement operator in DLs). Given a quasi-ordered search space $(\mathcal{DL}_{\mathcal{H}}, \sqsubseteq)$

- a downward refinement operator is a mapping $\rho : \mathcal{DL}_{\mathcal{H}} \rightarrow 2^{\mathcal{DL}_{\mathcal{H}}}$ such that

$$\forall C \in \mathcal{DL}_{\mathcal{H}} \quad \rho(C) \subseteq \{D \in \mathcal{DL}_{\mathcal{H}} \mid D \sqsubseteq C\}$$

- an upward refinement operator is a mapping $\delta : \mathcal{DL}_{\mathcal{H}} \rightarrow 2^{\mathcal{DL}_{\mathcal{H}}}$ such that

$$\forall C \in \mathcal{DL}_{\mathcal{H}} \quad \delta(C) \subseteq \{D \in \mathcal{DL}_{\mathcal{H}} \mid C \sqsubseteq D\}$$

Definition 3 (Refinement chain in DLs). Given a downward (resp., upward) refinement operator ρ (resp., δ) for a quasi-ordered search space $(\mathcal{DL}_{\mathcal{H}}, \sqsubseteq)$, a refinement chain from $C \in \mathcal{DL}_{\mathcal{H}}$ to $D \in \mathcal{DL}_{\mathcal{H}}$ is a sequence

$$C = C_0, C_1, \dots, C_n = D$$

such that $C_i \in \rho(C_{i-1})$ (resp., $C_i \in \delta(C_{i-1})$) for every $1 \leq i \leq n$.

Note that, given $(\mathcal{DL}, \sqsubseteq)$, there is an infinite number of generalizations and specializations. Usually one tries to define refinement operators that can traverse efficiently throughout the hypothesis space in pursuit of one of the correct definitions (w.r.t. the examples that have been provided).

Definition 4 (Properties of refinement operators in DLs). A downward refinement operator ρ for a quasi-ordered search space $(\mathcal{DL}_{\mathcal{H}}, \sqsubseteq)$ is

- (locally) finite iff $\rho(C)$ is finite for all concepts $C \in \mathcal{DL}_{\mathcal{H}}$.
- redundant iff there exists a refinement chain from a concept $C \in \mathcal{DL}_{\mathcal{H}}$ to a concept $D \in \mathcal{DL}_{\mathcal{H}}$, which does not go through some concept $E \in \mathcal{DL}_{\mathcal{H}}$ and a refinement chain from C to a concept equal to D , which does go through E .
- proper iff for all concepts $C, D \in \mathcal{DL}_{\mathcal{H}}$, $D \in \rho(C)$ implies $C \not\equiv D$.
- complete iff, for all concepts $C, D \in \mathcal{DL}_{\mathcal{H}}$ with $C \sqsubset D$, a concept $E \in \mathcal{DL}_{\mathcal{H}}$ with $E \equiv C$ can be reached from D by ρ .
- weakly complete iff, for all concepts $C \in \mathcal{DL}_{\mathcal{H}}$ with $C \sqsubset \top$, a concept $E \in \mathcal{DL}_{\mathcal{H}}$ with $E \equiv C$ can be reached from \top by ρ .

The corresponding notions for upward refinement operators are defined dually.

Designing a refinement operator needs to make decisions on which properties are most useful in practice regarding the underlying learning algorithm. Considering the properties reported in Def. 4, it has been shown that the most feasible property combination for Concept Learning in expressive DLs such as \mathcal{ALC} is $\{\text{weakly complete, complete, proper}\}$ [21]. Only for less expressive DLs like \mathcal{EL} , *ideal*, i.e. complete, proper and finite, operators exist [24].

4 Concept Learning as Constructive Reasoning in DLs

In this section, we formally characterize Concept Learning in DLs by emphasizing the constructive nature of the inductive inference.

4.1 Second-order Concept Expressions

We assume to start from the syntax of any Description Logic \mathcal{DL} where \mathbb{N}_c , \mathbb{N}_r , and \mathbb{N}_o are the alphabet of concept names, role names and individual names, respectively. In order to write second-order formulas, we introduce a set $\mathbb{N}_x = X_0, X_1, X_2, \dots$ of concept variables, which we can quantify over. We denote by \mathcal{DL}_X the language of concept terms obtained from \mathcal{DL} by adding \mathbb{N}_x .

Definition 5 (Concept term). A concept term in \mathcal{DL}_X is a concept formed according to the specific syntax rules of \mathcal{DL} augmented with the additional rule $C \rightarrow X$ for $X \in \mathbf{N}_x$.

Since we are not interested in second-order DLs as themselves, we restrict our language to particular existential second-order formulas of interest to this paper. In particular, we allow formulas involving an ABox. By doing so, we can easily model the computation of, e.g., the MSC, which was left out as future work in Colucci *et al.*'s framework. This paves the way to the modeling of Concept Learning as shown in the next subsection.

Definition 6 (Concept expression). Let $a_1, \dots, a_m \in \mathcal{DL}$ be individuals, $C_1, \dots, C_m, D_1, \dots, D_m \in \mathcal{DL}_X$ be concept terms containing concept variables X_0, X_1, \dots, X_n . A concept expression Γ in \mathcal{DL}_X is a conjunction

$$(C_1 \sqsubseteq D_1) \wedge \dots \wedge (C_l \sqsubseteq D_l) \wedge (C_{l+1} \not\sqsubseteq D_{l+1}) \wedge \dots \wedge (C_m \not\sqsubseteq D_m) \wedge (a_1 : D_1) \wedge \dots \wedge (a_l : D_l) \wedge (a_{l+1} : \neg D_{l+1}) \wedge \dots \wedge (a_m : \neg D_m) \quad (1)$$

of (negated or not) concept subsumptions and concept assertions with $1 \leq l \leq m$.

We use *General Semantics*, also called Henkin semantics, for interpreting concept variables [15]. In such a semantics, variables denoting unary predicates can be interpreted only by *some subsets* among all the ones in the powerset of the domain $2^{\Delta^{\mathcal{I}}}$ - instead, in Standard Semantics a concept variable could be interpreted as any subset of $\Delta^{\mathcal{I}}$. Adapting General Semantics to our problem, the structure we consider is exactly the sets interpreting concepts in \mathcal{DL} . That is, the interpretation $X^{\mathcal{I}}$ of a concept variable $X \in \mathcal{DL}_X$ must coincide with the interpretation $E^{\mathcal{I}}$ of some concept $E \in \mathcal{DL}$. The interpretations we refer to in the following definition are of this kind.

Definition 7 (Satisfiability). A concept expression Γ of the form (1) is satisfiable in \mathcal{DL} iff there exist $n + 1$ concepts $E_0, \dots, E_n \in \mathcal{DL}$ such that, extending the semantics of \mathcal{DL} for each interpretation \mathcal{I} , with: $(X_i)^{\mathcal{I}} = (E_i)^{\mathcal{I}}$ for $i = 0, \dots, n$, it holds that

1. for each $j = 1, \dots, l$, and every interpretation \mathcal{I} , $(C_j)^{\mathcal{I}} \subseteq (D_j)^{\mathcal{I}}$ and $(a_j)^{\mathcal{I}} \in (D_j)^{\mathcal{I}}$, and
2. for each $j = l + 1, \dots, m$, there exists an interpretation \mathcal{I} s.t. $(C_j)^{\mathcal{I}} \not\subseteq (D_j)^{\mathcal{I}}$ and $(a_j)^{\mathcal{I}} \notin (D_j)^{\mathcal{I}}$

Otherwise, Γ is said to be unsatisfiable in \mathcal{DL} .

Definition 8 (Solution). If a concept expression Γ of the form (1) is satisfiable in \mathcal{DL} , then $\langle E_0, \dots, E_n \rangle$ is a solution for Γ . Moreover, we say that the formula

$$\exists X_0 \dots \exists X_n. \Gamma \quad (2)$$

is true in \mathcal{DL} if there exist at least a solution for Γ , otherwise it is false.

4.2 Modeling Concept Learning with Second-Order DLs

It has been pointed out that the constructive reasoning tasks can be divided into two main categories: tasks for which we just need to compute a concept (or a set of concepts) and those for which we need to find a concept (or a set of concepts) according to some minimality/maximality criteria [8]. In the first case, we have a set of solutions while in the second one we also have a set of sub-optimal solutions to the main problem. E.g., the set of sub-optimal solutions in LCS is represented by the common subsumers. Both **MSC** and **Concept Learning** belong to this second category of constructive reasoning tasks. We remind the reader that **MSC** can be easily reduced to **LCS** for DLs that admit the one-of-concept constructor. However, this reduction is not trivial for the general case. Hereafter, first, we show how to model **MSC** in terms of formula (2). This step is to be considered as functional to the modeling of **Concept Learning**.

Most Specific Concept Intuitively, the **MSC** of individuals described in an ABox is a concept description that represents all the properties of the individuals including the concept assertions they occur in and their relationship to other individuals. Similar to the **LCS**, the **MSC** is uniquely determined up to equivalence. More precisely, the set of most specific concepts of individuals $a_1, \dots, a_k \in \mathcal{DL}$ forms an equivalence class, and if S is defined to be the set of all concept descriptions that have a_1, \dots, a_k as their instance, then this class is the least element in $[S]$ w.r.t. a partial ordering \preceq on equivalence classes induced by the quasi ordering \sqsubseteq . Analogously to the **LCS**, we refer to one of its representatives by $\text{MSC}(a_1, \dots, a_k)$. The **MSC** need not exist. Three different phenomena may cause the non existence of a least element in $[S]$, and thus, a **MSC**:

1. $[S]$ might be empty, or
2. $[S]$ might contain different minimal elements, or
3. $[S]$ might contain an infinite decreasing chain $[D_1] \succ [D_2] \dots$.

A concept E is not the **MSC** of a_1, \dots, a_k iff the following formula is true in \mathcal{DL} :

$$\exists X.(a_1 : X) \wedge \dots \wedge (a_k : X) \wedge (X \sqsubseteq E) \wedge (E \not\sqsubseteq X) \quad (3)$$

that is, E is not the **MSC** if there exists a concept X which is a most specific concept, and is strictly more specific than E .

Concept Learning Following Def. 1, we assume that $\text{Ind}_C^+(\mathcal{A}) = \{a_1, \dots, a_m\}$ and $\text{Ind}_C^-(\mathcal{A}) = \{b_1, \dots, b_n\}$. A concept $D \in \mathcal{DL}_{\mathcal{H}}$ is a correct concept definition for the target concept name C w.r.t. $\text{Ind}_C^+(\mathcal{A})$ and $\text{Ind}_C^-(\mathcal{A})$ iff it is a solution for the following second-order concept expression:

$$(C \sqsubseteq X) \wedge (X \sqsubseteq C) \wedge (a_1 : X) \wedge \dots \wedge (a_m : X) \wedge (b_1 : \neg X) \wedge \dots \wedge (b_n : \neg X) \quad (4)$$

The CSP version of the task is therefore modeled with the following formula.

$$\exists X.(C \sqsubseteq X) \wedge (X \sqsubseteq C) \wedge (a_1 : X) \wedge \dots \wedge (a_m : X) \wedge (b_1 : \neg X) \wedge \dots \wedge (b_n : \neg X) \quad (5)$$

A simple OP version of the task could be modeled with the formula:

$$\exists X.(C \sqsubseteq X) \wedge (X \sqsubseteq C) \wedge (X \sqsubseteq E) \wedge (E \not\sqsubseteq X) \wedge (a_1 : X) \wedge \dots \wedge (a_m : X) \wedge (b_1 : \neg X) \wedge \dots \wedge (b_n : \neg X) \quad (6)$$

which asks for solutions that are compliant with a minimality criterion involving concept subsumption checks. Therefore, a concept $E \in \mathcal{DL}_{\mathcal{H}}$ is not a correct concept definition for C w.r.t. $\text{Ind}_C^+(\mathcal{A})$ and $\text{Ind}_C^-(\mathcal{A})$ if there exists a concept X which is a most specific concept, and is strictly more specific than E .

5 Conclusions

In this paper, we have provided a formal characterization of **Concept Learning** in DLs according to a declarative modeling language which abstracts from the specific algorithms used to solve the task. To this purpose, we have defined a fragment of second-order logic under the general semantics which allows to express formulas involving concept assertions from an ABox. One such fragment enables us to cover the general case of MSC as well. Also, as a minor contribution, we have suggested that the *generalization as search* approach to **Concept Learning** in Mitchell's vision is just that unifying framework necessary for accompanying the declarative modeling language proposed in this paper with a way of computing solutions to the problems declaratively modeled with this language. More precisely, the computational method we refer to in this paper is based on the iterative application of suitable refinement operators. Since many refinement operators for DLs are already available in the literature, the method can be designed such that it can be instantiated with a refinement operator specifically defined for the DL in hand.

The preliminary results reported in this paper open a promising direction of research at the intersection of KR and ML. For this research we have taken inspiration from recent results in both areas. On one hand, Colucci *et al.*'s work provides a procedure which combines Tableaux calculi for DLs with rules for the substitution of concept variables in second-order concept expressions [8]. On the other hand, De Raedt *et al.*'s work shows that off-the-shelf constraint programming techniques can be applied to various ML problems, once reformulated as CSPs and OPs [9]. Interestingly, both works pursue a unified view on the inferential problems of interest to the respective fields of research. This match of research efforts in the two fields has motivated the work presented in this paper which, therefore, moves a step towards bridging the gap between KR and ML in areas such as the maintenance of KBs where the two fields have already produced interesting results though mostly independently from each other. New questions and challenges are raised by the cross-fertilization of these results. In the future, we intend to investigate how to express optimality criteria such as the information gain function within the second-order concept expressions and how the *generalization as search* approach can be effectively integrated with second-order calculus.

References

1. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Gottlob, G., Walsh, T. (eds.) IJCAI'03: Proceedings of the 18th International Joint Conference on Artificial intelligence. pp. 319–324. Morgan Kaufmann Publishers (2003)
2. Badea, L., Nienhuys-Cheng, S.: A refinement operator for description logics. In: Cussens, J., Frisch, A. (eds.) Inductive Logic Programming, Lecture Notes in Artificial Intelligence, vol. 1866, pp. 40–59. Springer-Verlag (2000)
3. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: Proc. of the 10th National Conf. on Artificial Intelligence. pp. 754–760. The AAAI Press / The MIT Press (1992)
4. Cohen, W.W., Hirsh, H.: Learnability of description logics. In: Haussler, D. (ed.) Proc. of the 5th Annual ACM Conf. on Computational Learning Theory. pp. 116–127. ACM (1992)
5. Cohen, W.W., Hirsh, H.: Learning the CLASSIC description logic: Theoretical and experimental results. In: Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'94). pp. 121–133. Morgan Kaufmann (1994)
6. Cohen, W.W., Hirsh, H.: The learnability of description logics with equality constraints. *Machine Learning* 17(2-3), 169–199 (1994)
7. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: Second-order description logics: Semantics, motivation, and a calculus. In: Haarslev, V., Toman, D., Weddell, G.E. (eds.) Proc. of the 23rd Int. Workshop on Description Logics (DL 2010). CEUR Workshop Proceedings, vol. 573. CEUR-WS.org (2010)
8. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: A unified framework for non-standard reasoning services in description logics. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI 2010 - 19th European Conference on Artificial Intelligence. *Frontiers in Artificial Intelligence and Applications*, vol. 215, pp. 479–484. IOS Press (2010)
9. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for data mining and machine learning. In: Fox, M., Poole, D. (eds.) Proc. of the 24th AAAI Conference on Artificial Intelligence. AAAI Press (2010)
10. Donini, F.M., Lenzerini, M., Nardi, D.: An efficient method for hybrid deduction. In: ECAI. pp. 246–252 (1990)
11. Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Knowledge-intensive induction of terminologies from metadata. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) The Semantic Web - ISWC 2004: Third International Semantic Web Conference. *Lecture Notes in Computer Science*, vol. 3298, pp. 441–455. Springer (2004)
12. Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Concept formation in expressive description logics. In: Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) *Machine Learning: ECML 2004*. *Lecture Notes in Computer Science*, vol. 3201, pp. 99–110. Springer (2004)
13. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Zelezný, F., Lavrač, N. (eds.) *Inductive Logic Programming*. *Lecture Notes in Computer Science*, vol. 5194, pp. 107–121. Springer (2008)
14. Frazier, M., Pitt, L.: CLASSIC learning. *Machine Learning* 25(2-3), 151–193 (1996)
15. Henkin, L.: Completeness in the theory of types. *Journal of Symbolic Logic* 15(2), 81–91 (1950)

16. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence* 26(2), 139–159 (2007)
17. Kietz, J.: Learnability of description logic programs. In: Matwin, S., Sammut, C. (eds.) *Inductive Logic Programming. Lecture Notes in Artificial Intelligence*, vol. 2583, pp. 117–132. Springer (2003)
18. Kietz, J.U., Morik, K.: A polynomial approach to the constructive induction of structural knowledge. *Machine Learning* 14(1), 193–217 (1994)
19. Küsters, R.: *Non-Standard Inferences in Description Logics, Lecture Notes in Computer Science*, vol. 2100. Springer (2001)
20. Küsters, R., Molitor, R.: Approximating most specific concepts in description logics with existential restrictions. *AI Communications* 15(1), 47–59 (2002)
21. Lehmann, J., Hitzler, P.: Foundations of refinement operators for description logics. In: Blockeel, H., Ramon, J., Shavlik, J.W., Tadepalli, P. (eds.) *Inductive Logic Programming. Lecture Notes in Artificial Intelligence*, vol. 4894, pp. 161–174. Springer (2008)
22. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. *Machine Learning* 78(1-2), 203–250 (2010)
23. Lehmann, J.: DL-Learner: Learning concepts in description logics. *Journal of Machine Learning Research* 10, 2639–2642 (2009)
24. Lehmann, J., Haase, C.: Ideal downward refinement in the \mathcal{EL} description logic. In: De Raedt, L. (ed.) *Inductive Logic Programming. Lecture Notes in Computer Science*, vol. 5989, pp. 73–87 (2010)
25. Lisi, F.A.: Building rules on top of ontologies for the semantic web with inductive logic programming. *Theory and Practice of Logic Programming* 8(03), 271–300 (2008)
26. Lisi, F.A.: Inductive logic programming in databases: From Datalog to $\mathcal{DL}+\log$. *Theory and Practice of Logic Programming* 10(3), 331–359 (2010)
27. McGuinness, D., Patel-Schneider, P.: Usability issues in knowledge representation systems. In: Mostow, J., Rich, C. (eds.) *Proc. of the 15th National Conf. on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference*. pp. 608–614. AAAI Press / The MIT Press (1998)
28. Michalski, R.S.: A theory and methodology of inductive learning. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) *Machine Learning: an artificial intelligence approach*, vol. I. Morgan Kaufmann (1983)
29. Mitchell, T.: Generalization as search. *Artificial Intelligence* 18, 203–226 (1982)
30. Mitchell, T.: *Machine Learning*. McGraw Hill (1997)
31. Muggleton, S.: Inductive logic programming. In: Arikawa, S., Goto, S., Ohsuga, S., Yokomori, T. (eds.) *Proc. of the 1st Conf. on Algorithmic Learning Theory*. Springer/Ohmsha (1990)
32. Nebel, B.: Reasoning and revision in hybrid representation systems, *Lecture Notes in Computer Science*, vol. 422. Springer (1990)
33. Nienhuys-Cheng, S., de Wolf, R.: Foundations of inductive logic programming, *Lecture Notes in Artificial Intelligence*, vol. 1228. Springer (1997)
34. Quinlan, J.: Learning logical definitions from relations. *Machine Learning* 5, 239–266 (1990)
35. Rouveirol, C., Ventos, V.: Towards learning in $CARIN-\mathcal{ALN}$. In: Cussens, J., Frisch, A. (eds.) *Inductive Logic Programming. Lecture Notes in Artificial Intelligence*, vol. 1866, pp. 191–208. Springer (2000)
36. Valiant, L.: A theory of the learnable. *Communications of the ACM* 27(11), 1134–1142 (1984)

Nonmonotonic Reasoning in Description Logic by Tableaux Algorithm with Blocking

Jaromír Malenko and Petr Štěpánek

Charles University, Malostranske namesti 25, 11800 Prague, Czech Republic,
Jaromir.Malenko@mff.cuni.cz, Petr.Stepanek@mff.cuni.cz

Abstract. To support nonmonotonic reasoning we introduce the description logic of minimal knowledge and negation as failure (MKNF-DL) as an extension of description logic with modal operators **K** and **A**. We discuss the problems with representation of a model for an MKNF-DL theory. For satisfiability checking of MKNF-DL theories, we introduce a tableaux algorithm with blocking, where blocking works with the modal part of an MKNF-DL theory. This blocking technique allows for reasoning about a larger class of MKNF-DL theories than previous approaches.

Recently, Description Logics (DL) are used to represent and reason about knowledge bases (KBs). In practical applications, the monotonic property of standard logics, which includes DLs, may be undesirable.

Hence, we introduce $\mathcal{ALCK}_{\mathcal{NF}}$, the DL of minimal knowledge and negation as failure (MKNF-DL) as an extension of description logic (DL) with modal operators **K** and **A**. Advanced reasoning applications, including epistemic queries, integrity constraints and default rules can be represented by $\mathcal{ALCK}_{\mathcal{NF}}$ [4].

Next, we introduce a reasoning technique for $\mathcal{ALCK}_{\mathcal{NF}}$. To this end the representation of $\mathcal{ALCK}_{\mathcal{NF}}$ models is crucial. The models of $\mathcal{ALCK}_{\mathcal{NF}}$ are not first-order representable. Hence, we define *subjectively quantified* KBs which are representable by \mathcal{ALC} theory. However, this \mathcal{ALC} theory may be infinite. Previous approaches [1, 3] defined *simple* KBs, a subset of subjectively quantified KBs, which are representable by finite \mathcal{ALC} theory. The intention of our research was an effective reasoning method for subjectively quantified KB. We achieved this by introducing a tableaux algorithm with blocking; however, for the algorithm to be complete, a further restriction to *minimality-proper* KBs is necessary. Still, minimality-proper KBs include simple KBs.

1 Basic Formalism

We start with usual formalism of MKNF-DL, which roughly follow Donini [1].

1.1 Syntax

Definition 1 ($\mathcal{ALCK}_{\mathcal{NF}}$ Syntax) *The $\mathcal{ALCK}_{\mathcal{NF}}$ syntax is defined as follows:*

$$\begin{aligned} C &::= \top \mid \perp \mid C_a \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \mathbf{K}C \mid \mathbf{A}C \\ R &::= R_a \mid \mathbf{K}R \mid \mathbf{A}R \end{aligned}$$

where C_a is an atomic concept, C, C_1, C_2 are concept expressions, R_a is an atomic role, R is a role, and \mathbf{K} and \mathbf{A} are modal operators.

Definition 2 An $\mathcal{ALCK}_{\mathcal{NF}}$ concept C is subjective if each \mathcal{ALC} atomic subconcept of C is also a subconcept of a modal subconcept of C . An $\mathcal{ALCK}_{\mathcal{NF}}$ concept C is objective if C does not contain a modal role or a modal subconcept. An $\mathcal{ALCK}_{\mathcal{NF}}$ concept C is mixed if C is neither subjective nor objective.

Objective concepts are the \mathcal{ALC} concepts. For example, concept $\exists\mathbf{K}R.C$ is neither subjective nor objective.

The $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base (KB) is an extension of \mathcal{ALC} KB to which we add the modal formulae.

Definition 3 ($\mathcal{ALCK}_{\mathcal{NF}}$ Knowledge Base) The $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base Σ is a triple $\langle \mathcal{T}, \Gamma, \mathcal{A} \rangle$, where $TBox \mathcal{T}$ is a set of objective axioms, $MBox \Gamma$ is a set of non-objective axioms, and $ABox \mathcal{A}$ is a set of (both objective and non-objective) assertions.

In the remainder we consider the following simplification of notation. We assume an $\mathcal{ALCK}_{\mathcal{NF}}$ KB $\Sigma = \langle \mathcal{T}, \Gamma, \mathcal{A} \rangle$. \mathbf{M} denotes a modal operator, i.e., $\mathbf{M} \in \{\mathbf{K}, \mathbf{A}\}$, \mathbf{N} denotes a possibly negated modal operator, i.e., $\mathbf{N} \in \{\mathbf{K}, \mathbf{A}, \neg\mathbf{K}, \neg\mathbf{A}\}$. R_a denotes an atomic \mathcal{ALC} role, C_a, D_a denote atomic \mathcal{ALC} concepts, A, B, C, D (possibly with indices or primes) denote arbitrary $\mathcal{ALCK}_{\mathcal{NF}}$ concepts.

1.2 Semantics

Remind, that an \mathcal{ALC} interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ consists of domain Δ and interpretation function $\cdot^{\mathcal{I}}$ which maps each atomic concept C_a to a subset of domain: $C_a^{\mathcal{I}} \subseteq \Delta$, each atomic role R_a to a subset of cartesian product of domain: $R_a^{\mathcal{I}} \subseteq \Delta \times \Delta$, and each individual name x to an element of domain: $x^{\mathcal{I}} \in \Delta$.

The semantics of $\mathcal{ALCK}_{\mathcal{NF}}$ considers sets of \mathcal{ALC} interpretations with the following properties:

- (i) The domain Δ is a countable (possibly infinite) set.
- (ii) All \mathcal{ALC} interpretations are defined over the same domain Δ .
- (iii) Each individual name in every interpretation maps to the same domain element.

Definition 4 ($\mathcal{ALCK}_{\mathcal{NF}}$ Semantics) An $\mathcal{ALCK}_{\mathcal{NF}}$ interpretation is a triple $(\mathcal{I}, \mathcal{M}, \mathcal{N})$, where \mathcal{I} is an \mathcal{ALC} interpretation $(\Delta, \cdot^{\mathcal{I}})$, and \mathcal{M} and \mathcal{N} are sets of \mathcal{ALC} interpretations.

Atomic concepts C_a , roles R_a , and individuals a are interpreted in \mathcal{I} as usual in \mathcal{ALC} :

$$\begin{aligned} (C_a)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= C_a^{\mathcal{I}} \\ (R_a)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= R_a^{\mathcal{I}} \\ (a)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= a^{\mathcal{I}}. \end{aligned}$$

The $\mathcal{ALCK}_{\mathcal{NF}}$ interpretation $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ is extended to non-atomic and modal concepts and modal roles as follows:

$$\begin{aligned}
(\top)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \Delta \\
(\perp)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \emptyset \\
(\neg C)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \Delta \setminus (C)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \\
(C_1 \sqcap C_2)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= (C_1)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \cap (C_2)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \\
(C_1 \sqcup C_2)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= (C_1)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \cup (C_2)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \\
(\exists R.C)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \{x \in \Delta \mid \exists y \in \Delta : (x, y) \in (R)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \wedge y \in (C)^{\mathcal{I}, \mathcal{M}, \mathcal{N}}\} \\
(\forall R.C)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \{x \in \Delta \mid \forall y \in \Delta : (x, y) \in (R)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \Rightarrow y \in (C)^{\mathcal{I}, \mathcal{M}, \mathcal{N}}\} \\
(\mathbf{K}C)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \bigcap_{\mathcal{J} \in \mathcal{M}} (C)^{\mathcal{J}, \mathcal{M}, \mathcal{N}} \\
(\mathbf{A}C)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \bigcap_{\mathcal{J} \in \mathcal{N}} (C)^{\mathcal{J}, \mathcal{M}, \mathcal{N}} \\
(\mathbf{K}R_a)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \bigcap_{\mathcal{J} \in \mathcal{M}} (R_a)^{\mathcal{J}, \mathcal{M}, \mathcal{N}} \\
(\mathbf{A}R_a)^{\mathcal{I}, \mathcal{M}, \mathcal{N}} &= \bigcap_{\mathcal{J} \in \mathcal{N}} (R_a)^{\mathcal{J}, \mathcal{M}, \mathcal{N}}
\end{aligned}$$

Definition 5 (Satisfiability in $(\mathcal{I}, \mathcal{M}, \mathcal{N})$) A concept inclusion $C \sqsubseteq D$ is satisfied in $(\mathcal{I}, \mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{I}, \mathcal{M}, \mathcal{N}) \models C \sqsubseteq D$, iff $C^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \subseteq D^{\mathcal{I}, \mathcal{M}, \mathcal{N}}$. A concept assertion $C(a)$ is satisfied in $(\mathcal{I}, \mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{I}, \mathcal{M}, \mathcal{N}) \models C(a)$, iff $a \in C^{\mathcal{I}, \mathcal{M}, \mathcal{N}}$. A role assertion $R(a, b)$ is satisfied in $(\mathcal{I}, \mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{I}, \mathcal{M}, \mathcal{N}) \models R(a, b)$, iff $(a, b) \in R^{\mathcal{I}, \mathcal{M}, \mathcal{N}}$.

Definition 6 (Satisfiability in $(\mathcal{M}, \mathcal{N})$) A concept inclusion $C \sqsubseteq D$ is satisfied in $(\mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{M}, \mathcal{N}) \models C \sqsubseteq D$, iff $C^{\mathcal{I}, \mathcal{M}, \mathcal{N}} \subseteq D^{\mathcal{I}, \mathcal{M}, \mathcal{N}}$ for each $\mathcal{I} \in \mathcal{M}$. A concept assertion $C(a)$ is satisfied in $(\mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{M}, \mathcal{N}) \models C(a)$, iff $a \in C^{\mathcal{I}, \mathcal{M}, \mathcal{N}}$ for each $\mathcal{I} \in \mathcal{M}$. A role assertion $R(a, b)$ is satisfied in $(\mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{M}, \mathcal{N}) \models R(a, b)$, iff $(a, b) \in R^{\mathcal{I}, \mathcal{M}, \mathcal{N}}$ for each $\mathcal{I} \in \mathcal{M}$.

A TBox \mathcal{T} is satisfied in $(\mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{M}, \mathcal{N}) \models \mathcal{T}$, iff all axioms in \mathcal{T} are satisfied in $(\mathcal{M}, \mathcal{N})$. An MBox Γ is satisfied in $(\mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{M}, \mathcal{N}) \models \Gamma$, iff all axioms in Γ are satisfied in $(\mathcal{M}, \mathcal{N})$. An ABox \mathcal{A} is satisfied in $(\mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{M}, \mathcal{N}) \models \mathcal{A}$, iff all assertions in \mathcal{A} are satisfied in $(\mathcal{M}, \mathcal{N})$. A KB Σ is satisfied in $(\mathcal{M}, \mathcal{N})$, denoted as $(\mathcal{M}, \mathcal{N}) \models \Sigma$, iff all \mathcal{T} , Γ and \mathcal{A} are satisfied in $(\mathcal{M}, \mathcal{N})$.

Up to now, the definition of modal operators \mathbf{K} and \mathbf{A} were the same. Their meaning is distinguished by imposing the maximality condition on \mathbf{K} .

Definition 7 ($\mathcal{ALCK}_{\mathcal{NF}}$ Model) A set of \mathcal{ALC} interpretations \mathcal{M} is a model for an $\mathcal{ALCK}_{\mathcal{NF}}$ KB Σ iff the following two conditions hold:

- (i) the structure $(\mathcal{M}, \mathcal{M})$ satisfies Σ ; and
- (ii) for each set of \mathcal{ALC} interpretations \mathcal{M}' , if $\mathcal{M}' \supset \mathcal{M}$ then the structure $(\mathcal{M}', \mathcal{M})$ does not satisfy Σ .

Definition 8 (Entailment) A concept inclusion $C \sqsubseteq D$ is a consequence of KB Σ , denoted as $\Sigma \models C \sqsubseteq D$, iff $(\mathcal{M}, \mathcal{M}) \models C \sqsubseteq D$ for each model \mathcal{M} of Σ . A concept assertion $C(a)$ is a consequence of KB Σ , denoted as $\Sigma \models$

$C(a)$, iff $(\mathcal{M}, \mathcal{M}) \models C(a)$ for each model \mathcal{M} of Σ . A role assertion $R(a, b)$ is a consequence of KB Σ , denoted as $\Sigma \models R(a, b)$, iff $(\mathcal{M}, \mathcal{M}) \models R(a, b)$ for each model \mathcal{M} of Σ .

Definition 9 (Satisfiability) A KB Σ is satisfiable if there is a model for Σ . Two KBs Σ and Σ' are equivalent, denoted as $\Sigma \equiv \Sigma'$, iff for every set of \mathcal{ALC} interpretations \mathcal{M} , \mathcal{M} is model for Σ iff \mathcal{M} is model for Σ' . Two concepts C and C' are equivalent in KB Σ , denoted as $\Sigma \models C \equiv C'$, iff for every set of \mathcal{ALC} interpretations \mathcal{M} of Σ , $\mathcal{M} \models C \sqsubseteq C'$ and $\mathcal{M} \models C' \sqsubseteq C$.

2 Model Representation

Our intention is to introduce a reasoning technique for $\mathcal{ALCK}_{\mathcal{NF}}$. This will be attained in the next section. In this section, we discuss the *representation* of $\mathcal{ALCK}_{\mathcal{NF}}$ models.

In our treatment of representing and reasoning in $\mathcal{ALCK}_{\mathcal{NF}}$ we follow several approaches to reasoning in propositional modal logic [2, 5, 6].

The reasoning problem for $\mathcal{ALCK}_{\mathcal{NF}}$ is reduced to several reasoning problems in the underlying nonmodal logic \mathcal{ALC} . Each model for an $\mathcal{ALCK}_{\mathcal{NF}}$ KB Σ is characterized by an \mathcal{ALC} KB. Therefore, the set of \mathcal{ALC} KBs that represents all the models of Σ constitute a nonmodal representation of Σ . Such a representation allows for using classical reasoning techniques for \mathcal{ALC} to solve the reasoning problems for $\mathcal{ALCK}_{\mathcal{NF}}$.

Recall that an $\mathcal{ALCK}_{\mathcal{NF}}$ model \mathcal{M} is a set of \mathcal{ALC} interpretations. The model \mathcal{M} is *first-order representable* if there exists a first-order theory (resp. \mathcal{ALC} KB) $\Sigma_{\mathcal{M}}$ such that

$$\mathcal{M} = \{\mathcal{I} : \mathcal{I} \models \Sigma_{\mathcal{M}}\}.$$

Therefore the set of interpretations belonging to \mathcal{M} can be represented by a first-order theory $\Sigma_{\mathcal{M}}$. Note that the theory $\Sigma_{\mathcal{M}}$ may be either finite or infinite.

Following the motivation of this approach, we consider only the case when the $\mathcal{ALCK}_{\mathcal{NF}}$ model \mathcal{M} is \mathcal{ALC} *representable*, i.e. there exists an \mathcal{ALC} KB $\Sigma_{\mathcal{M}}$ such that $\mathcal{M} = \{\mathcal{I} : \mathcal{I} \models \Sigma_{\mathcal{M}}\}$. Moreover, we consider only the case when the corresponding $\Sigma_{\mathcal{M}}$ is finite. Since \mathcal{ALC} is a fragment of first-order logic, if \mathcal{M} is \mathcal{ALC} representable then \mathcal{M} is also first-order representable.

To guarantee \mathcal{ALC} representability of $\mathcal{ALCK}_{\mathcal{NF}}$ KB models, the KB is restricted to a *subjectively quantified KB*. In previous publications [1, 3], a subjectively quantified KB is further restricted to *simple KB* whose models admit a representation in terms of *finite* \mathcal{ALC} KB and guarantee decidability of reasoning (specifically termination of the tableaux algorithm). This progress is depicted in Figure 1.

The approach of subjectively quantified KBs and simple KBs was first introduced and extensively analysed in Donini et al. [1]. A further research was done by Ke et al. [3]. We now compare both papers and explain our approach.

First, Donini et al. [1] claim that the models of $\mathcal{ALCK}_{\mathcal{NF}}$ KBs cannot be characterized by first-order theories.

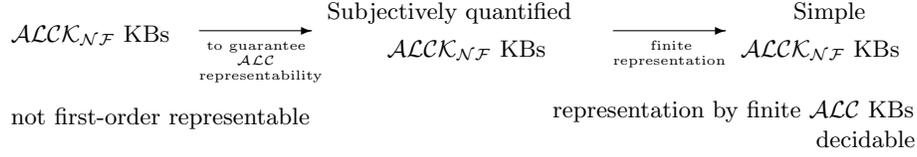


Fig. 1. The structure of work on $\mathcal{ALCK}_{\mathcal{NF}}$ model representation. In previous publications [1, 3], to obtain a reasoning algorithm, the language of KBs was syntactically restricted to subjectively quantified and then to simple KB. In this thesis, we provide a reasoning algorithm that works with subjectively quantified KBs. Since our approach does not require the simple restriction, it allows reasoning over greater set of KBs.

Theorem 10 *The models of $\mathcal{ALCK}_{\mathcal{NF}}$ are not first-order representable.*

Briefly, the proof considers a carefully constructed $\mathcal{ALCK}_{\mathcal{NF}}$ KB Σ , shows a model \mathcal{M} for Σ , and shows that \mathcal{M} cannot be characterized in terms of first-order theory.

Second, they identify a subset of $\mathcal{ALCK}_{\mathcal{NF}}$ KBs whose models are first-order representable by means of (either finite or infinite) first-order theories. This is achieved by the notion of subjectively quantified KBs. We consider two definitions from previous papers [1, 3].

Definition 11 (Subjectively Quantified KB) *A subjectively quantified $\mathcal{ALCK}_{\mathcal{NF}}$ KB is an $\mathcal{ALCK}_{\mathcal{NF}}$ KB Σ such that each concept C of the form $\exists R.D$ or $\forall R.D$ satisfies one of the following conditions:*

- (i) *R is an \mathcal{ALCC} role and D is an \mathcal{ALCC} concept;*
- (ii) *R is of the form \mathbf{MR}_a and D is subjective.*

To summarize, the Definition 11 by [3] is more general than similar definition by [1], and we use it for the rest of this paper. However the difference between the definitions has no implications for the tableaux algorithm, since the algorithm considers the flatten normal forms of formulae (created in preprocessing stage).

Theorem 12 *A subjectively quantified $\mathcal{ALCK}_{\mathcal{NF}}$ KB is representable by an (either finite or infinite) \mathcal{ALCC} theory.*

Briefly, the proof shows that the models of a subjectively quantified $\mathcal{ALCK}_{\mathcal{NF}}$ KB can be characterized in terms of an \mathcal{ALCC} KB. The characterization relies on the notion of *modal atom*: there is a one-to-one correspondence between certain sets of modal atoms and the models of subjectively quantified $\mathcal{ALCK}_{\mathcal{NF}}$ KB.

2.1 Previous Work

In previous publications [1, 3], authors further restrict $\mathcal{ALCK}_{\mathcal{NF}}$ KBs to obtain a finite characterization of models (that are also first-order representable). This is achieved by the notion of simple KBs.

In the following, we say that an $\mathcal{ALCK}_{\mathcal{NF}}$ concept is *simple* if C is subjectively quantified and each quantified concept subexpression of the form $\exists \mathbf{A}R.ND$, $\forall \mathbf{A}R.ND$, where $\mathbf{N} \in \{\mathbf{K}, \neg\mathbf{K}, \mathbf{A}, \neg\mathbf{A}\}$, occurring in C is such that in D there are no occurrences of role expressions of the form \mathbf{KR} .

Definition 13 (Simple KB) *A simple $\mathcal{ALCK}_{\mathcal{NF}}$ KB $\Sigma = \langle \mathcal{T}, \Gamma, \mathcal{A} \rangle$ is an $\mathcal{ALCK}_{\mathcal{NF}}$ KB that satisfies the following conditions:*

- (i) Γ is a set of $\mathcal{ALCK}_{\mathcal{NF}}$ simple inclusions, i.e. inclusions assertions of the form $\mathbf{KC} \sqsubseteq D$, where C is an \mathcal{ALC} -concept such that $\mathcal{T} \not\models \top \sqsubseteq C$, and D is a subjectively quantified concept expression in which there are no occurrences of the operator \mathbf{K} within the scope of quantifiers; and
- (ii) \mathcal{A} is a set of instance assertions such that all concept subexpressions occurring in \mathcal{A} are simple.

Ke et al. [3] significantly loosens the definition of a simple KB by allowing more concept assertions in \mathcal{A} . Both [1, 3] conclude with a proof of the following theorem.

Theorem 14 *A simple $\mathcal{ALCK}_{\mathcal{NF}}$ KB is representable by a finite \mathcal{ALC} theory.*

2.2 Our Approach

We present a reasoning algorithm for deciding the satisfiability of a subjectively quantified $\mathcal{ALCK}_{\mathcal{NF}}$ KB. We use a novel tableaux algorithm with a blocking technique to deal with modal part of $\mathcal{ALCK}_{\mathcal{NF}}$ theory.

In standard DLs, the tableaux algorithm tests satisfiability of a concept description. The algorithm starts with a concept description and applies consistency-preserving expansion rules that build a tree representing a model. In the tree, the nodes correspond to individuals and are labelled with sets of DL concept descriptions the individual belongs to.

The models of standard DL KBs can be infinite and therefore the tableaux algorithm uses a blocking technique that identifies infinite branches. This allows to represent the infinite models by finitely-representable models.

The main motivation of our approach is analogous: the the infinite models of $\mathcal{ALCK}_{\mathcal{NF}}$ can be blocked and represented by finitely-representable $\mathcal{ALCK}_{\mathcal{NF}}$ models.

3 Reasoning by Tableaux Algorithm with Blocking

In our approach, we reduce the reasoning problem for $\mathcal{ALCK}_{\mathcal{NF}}$ to several reasoning problems in \mathcal{ALC} . Since each model for a subjectively quantified $\mathcal{ALCK}_{\mathcal{NF}}$ KB Σ can be represented by a set of \mathcal{ALC} KBs, this allows us to use classical reasoning techniques for \mathcal{ALC} to solve reasoning problems in $\mathcal{ALCK}_{\mathcal{NF}}$.

In this section we define the tableaux algorithm that tests the satisfiability of a subjectively quantified $\mathcal{ALCK}_{\mathcal{NF}}$ KB $\Sigma = \langle \mathcal{T}, \Gamma, \mathcal{A} \rangle$.

The reader is referred to Malenko et al. [4] for detailed treatment and proofs of theorems in this section.

Since every $\mathcal{ALCK}_{\mathcal{NF}}$ KB can be translated to flatten normal form, we assume w.l.o.g. the Σ to be in flatten normal form.

The satisfiability problem for Σ is solved by a tableaux algorithm for Σ . The tableaux algorithm is trying to construct a model for Σ by means of branches which correspond to finite subsets of modal atoms. A branch satisfying certain conditions corresponds to a model for Σ .

The $\mathcal{ALCK}_{\mathcal{NF}}$ tableaux algorithm is similar to a standard \mathcal{ALC} tableaux algorithm. However, there are differences: Only modal assertions are decomposed by the $\mathcal{ALCK}_{\mathcal{NF}}$ tableaux algorithm, including the GCIs in Γ . An \mathcal{ALC} reasoner is used as an underlying reasoner, which considers \mathcal{T} and \mathcal{ALC} assertions through the so-called objective knowledge of a branch. The \mathcal{ALC} reasoner is called in the conditions of trigger-rule and testing the conditions of models (open branch, pre-preferred branch, minimality condition).

A *branch* \mathcal{B} is a set of membership assertions of the form $C(x)$, where C is an $\mathcal{ALCK}_{\mathcal{NF}}$ concept description.

The tableaux starts with an *initial branch* $\mathcal{B}_0 = \{\mathbf{K}C(x) \mid C(x) \in \mathcal{A}\}$. New branches are obtained from the current branch by applying the tableaux expansion rules from Figure 2. We use $\neg C$ to denote the negation normal form of the $\neg C$. The set of individuals appearing in \mathcal{B} is denoted by $\mathcal{O}_{\mathcal{B}}$.

We now briefly describe the $\mathcal{ALCK}_{\mathcal{NF}}$ expansion rules and compare them with standard DL tableaux rules [1]:

- The \sqcap -rule is analogous to the corresponding DL tableaux rule.
- The \sqcup -rule adds both disjuncts (as they are or possibly negated) to the tableaux, because they are needed in the minimality check. The parts of the rule consider all the cases that can happen; the part (d) detect and inconsistency and enforces a clash in the underlying DL reasoner. In the corresponding DL tableaux rule, either of the conjunct is added to the tableaux.
- The \exists -rule is analogous to the corresponding DL tableaux rule.
- The \forall -rule has two parts. The first part deals with modal roles and is analogous to DL tableaux rule. The second part deals with the relationship between $\mathbf{K}R(x, y)$ and $\mathbf{A}R(x, y)$. In the tableaux algorithm, the “value” of $\mathbf{A}R(x, y)$ is by default “false” until the appearance of $\mathbf{K}R(x, y)$ supporting it to be “true” (this is possible since $\mathbf{A}R(x, y) \in MA_{\Delta}(\Sigma)$ it is supported by $\forall \mathbf{A}R.C(x) \in MA_{\Delta}(\Sigma)$).
- The trigger-rule takes Γ into account. The underlying DL reasoner is used to check that $C(a)$ is entailed in \mathbf{K} -objective knowledge, which represents “what is known so far” by \mathcal{T} and \mathcal{B} .

To employ a blocking technique, the algorithm keeps ordering of individual names according to time of their introduction to tableaux algorithm. Moreover, for each individual y that is introduced into the tableaux by application of the \exists -rule to individual x , the algorithm keeps track of the relationship and we define $parent(y) = x$. Then, $predecessor$ is transitive closure of $parent$.

- \sqcap -rule:** If $C \sqcap D(x) \in \mathcal{B}$,
and $\{C(x), D(x)\} \not\subseteq \mathcal{B}$,
then add $C(x), D(x)$ to \mathcal{B} .
- \sqcup -rule:** If $C \sqcup D(x) \in \mathcal{B}$ we distinguish the following four cases:
(a) if $\{C(x), \neg C(x), D(x), \neg D(x)\} \cap \mathcal{B} = \emptyset$,
then add either $C(x), D(x)$ or $C(x), \neg D(x)$ or $\neg C(x), D(x)$ to \mathcal{B} .
(b) if $\{C(x), \neg C(x)\} \cap \mathcal{B} = \emptyset$ and $\{D(x), \neg D(x)\} \cap \mathcal{B} \neq \emptyset$,
then add either $C(x)$ or $\neg C(x)$ to \mathcal{B} .
(c) if $\{C(x), \neg C(x)\} \cap \mathcal{B} \neq \emptyset$ and $\{D(x), \neg D(x)\} \cap \mathcal{B} = \emptyset$,
then add either $D(x)$ or $\neg D(x)$ to \mathcal{B} .
(d) if $\{\neg C(x), \neg D(x)\} \subseteq \mathcal{B}$,
then add $C(x)$ to \mathcal{B} .
- \exists -rule:** If $\exists \mathbf{MR}.C(x) \in \mathcal{B}$,
and $\{\mathbf{MR}(x, y), C(y)\} \not\subseteq \mathcal{B}$ for any $y \in \mathcal{O}_{\mathcal{B}}$,
and x is not blocked by some $y \in \mathcal{O}_{\mathcal{B}}$,
then add $\mathbf{MR}(x, z), C(z)$ to \mathcal{B} , for some $z \in \mathcal{O}_{\mathcal{B}} \cup \{i\}$, where $i \notin \mathcal{O}_{\mathcal{B}}$.
- \forall -rule:** We distinguish the following two cases:
(a) if $\forall \mathbf{MR}.C(x) \in \mathcal{B}$,
then for each $\mathbf{MR}(x, y) \in \mathcal{B}$, if $C(y) \notin \mathcal{B}$ then add $C(y)$ to \mathcal{B} .
(b) if $\forall \mathbf{AR}.C(x) \in \mathcal{B}$,
then for each $\mathbf{KR}(x, y) \in \mathcal{B}$, if $\mathbf{AR}(x, y) \notin \mathcal{B}$ then add $\mathbf{AR}(x, y)$ to \mathcal{B} .
- trigger-rule:** If $\mathbf{KC}_a \sqsubseteq D \in \Gamma, x \in \mathcal{O}_{\mathcal{B}}, \text{Ob}_{\mathbf{K}}(\mathcal{B}) \models C_a(x)$,
and $\{\mathbf{KC}_a(x), D(x)\} \not\subseteq \mathcal{B}$,
then add $\mathbf{KC}_a(x), D(x)$ to \mathcal{B} .

Fig. 2. Expansion rules for the tableaux algorithm of $\mathcal{ALCK}_{\mathcal{NF}}$

Definition 15 (Blocking) *The application of an expansion rule to an individual x is blocked in a branch \mathcal{B} if there is a predecessor y of x and a predecessor z of y such that $\{C \mid C(y) \in \mathcal{B}\} = \{C \mid C(z) \in \mathcal{B}\}$.*

The idea behind blocking is that the blocked individual x can use role successors of y instead of generating new ones. The ordering of individual names according to time of their introduction to tableaux algorithm is considered to avoid cyclic blocking (of x by y and vice versa).

To guarantee termination of this blocking technique, the following strategy for application of expansion rules to the individuals in a branch is used: all rules are applied to the first introduced individual until no more rules apply, then all rules are applied to the second introduced individual until no more rules apply, etc. When a rule is successfully applied to an individual, the rule applications start again from the first individual. When no expansion rules are applicable the tableaux algorithm ends. An effective implementation of this strategy can be achieved by keeping track of tableaux changes and proper backtracking (back-jumping).

Definition 16 (Completed Branch) *We say a branch \mathcal{B} is completed if no expansion rules from Figure 2 are applicable to \mathcal{B} .*

Definition 17 ($(P_{\mathcal{B}}, N_{\mathcal{B}})$) *The partition $(P_{\mathcal{B}}, N_{\mathcal{B}})$ associated with a branch \mathcal{B} is defined as follows:*

$$\begin{aligned} P_{\mathcal{B}} &= \{\mathbf{MC}(x) \mid \mathbf{MC}(x) \in \mathcal{B}\} \cup \{\mathbf{MR}(x, y) \mid \mathbf{MR}(x, y) \in \mathcal{B}\} \\ N_{\mathcal{B}} &= \{\mathbf{MC}(x) \mid \neg \mathbf{MC}(x) \in \mathcal{B}\} \end{aligned}$$

Here, $P_{\mathcal{B}}$ is a set of atoms which are true, $N_{\mathcal{B}}$ is a set of atoms which are false. Atoms not in $P_{\mathcal{B}} \cup N_{\mathcal{B}}$ are assumed to be false.

Definition 18 (Objective Knowledge) *Let \mathcal{B} be a branch for Σ . The \mathcal{ALC} KB*

$$\begin{aligned} Ob_{\mathbf{K}}(\mathcal{B}) &= \langle \mathcal{T}, \{C(x) \mid \mathbf{KC}(x) \in P_{\mathcal{B}}\} \cup \{R_a(x, y) \mid \mathbf{KR}_a(x, y) \in P_{\mathcal{B}}\} \rangle \\ Ob_{\mathbf{A}}(\mathcal{B}) &= \langle \mathcal{T}, \{C(x) \mid \mathbf{AC}(x) \in P_{\mathcal{B}}\} \cup \{R_a(x, y) \mid \mathbf{AR}_a(x, y) \in P_{\mathcal{B}}\} \rangle \end{aligned}$$

is called \mathbf{K} -objective, resp. \mathbf{A} -objective, knowledge for \mathcal{B} .

The objective knowledge is an \mathcal{ALC} KB passed to an underlying \mathcal{ALC} reasoner to check a certain set of conditions.

Definition 19 (Open Branch) *A branch \mathcal{B} is open if all of the following conditions hold:*

- (i) $Ob_{\mathbf{K}}(\mathcal{B})$ is satisfiable;
- (ii) $Ob_{\mathbf{A}}(\mathcal{B})$ is satisfiable;
- (iii) $Ob_{\mathbf{K}}(\mathcal{B}) \not\models C(x)$ for each $\mathbf{KC}(x) \in N_{\mathcal{B}}$;
- (iv) $Ob_{\mathbf{A}}(\mathcal{B}) \not\models C(x)$ for each $\mathbf{AC}(x) \in N_{\mathcal{B}}$.

If there is an open branch \mathcal{B} for Σ , then there is an $\mathcal{ALCK}_{\mathcal{NF}}$ structure $(\mathcal{M}, \mathcal{N})$ that satisfies Σ .

Definition 20 (Pre-preferred Branch) *A branch \mathcal{B} is pre-preferred if all of the following conditions hold:*

- (i) \mathcal{B} is completed and open;
- (ii) $Ob_{\mathbf{K}}(\mathcal{B}) \models Ob_{\mathbf{A}}(\mathcal{B})$;
- (iii) $Ob_{\mathbf{K}}(\mathcal{B}) \not\models C(x)$ for each $\mathbf{AC}(x) \in N_{\mathcal{B}}$.

If there is a pre-preferred branch \mathcal{B} for Σ , then there is an $\mathcal{ALCK}_{\mathcal{NF}}$ structure $(\mathcal{M}, \mathcal{M})$ that satisfies Σ .

The minimality condition below checks whether the set of interpretations $\mathcal{M} = \{\mathcal{I} \mid \mathcal{I} \models Ob_{\mathbf{K}}(\mathcal{B})\}$ represents a model for Σ . The minimality condition checks the minimality of a branch up to the renaming of individuals introduced by the \exists -rule.

For a branch \mathcal{B} and an injection f , the branch obtained from \mathcal{B} by replacing each occurrence of individual x by $f(x)$ for each $x \in \mathcal{O}_{\Sigma} \setminus \mathcal{O}_{\mathcal{B}}$, is called a *renamed branch* of \mathcal{B} and denoted $f(\mathcal{B})$.

Definition 21 (Minimality Condition) Let \mathcal{B} be a completed branch for Σ . \mathcal{B} satisfies the minimality condition if there does not exist a completed and open branch \mathcal{B}' for Σ and an injection $f : \mathcal{O}_{\mathcal{B}'} \setminus \mathcal{O}_{\Sigma} \rightarrow \mathcal{O}_{\mathcal{B}} \setminus \mathcal{O}_{\Sigma}$ such that $|\mathcal{O}_{\mathcal{B}'}| \leq |\mathcal{O}_{\mathcal{B}}|$ and all of the following conditions hold:

- (i) $Ob_{\mathbf{K}}(\mathcal{B}) \models Ob_{\mathbf{K}}(f(\mathcal{B}'))$;
- (ii) $Ob_{\mathbf{K}}(f(\mathcal{B}')) \not\models Ob_{\mathbf{K}}(\mathcal{B})$;
- (iii) $Ob_{\mathbf{K}}(\mathcal{B}) \models Ob_{\mathbf{A}}(f(\mathcal{B}'))$;
- (iv) $Ob_{\mathbf{K}}(\mathcal{B}) \not\models C(x)$ for each $\mathbf{A}C(x) \in N_{f(\mathcal{B}')}$.

If minimality condition does not hold, there exists a structure $(\mathcal{M}', \mathcal{M})$ such that $\mathcal{M}' \supset \mathcal{M}$ and $(\mathcal{M}', \mathcal{M})$ satisfies Σ . This implies that \mathcal{M} is not a model for Σ .

For the tableaux algorithm with this minimality condition to be complete [3], KBs must be restricted according to the following definition.

Definition 22 (Minimality-proper KB) An $\mathcal{ALCK}_{\mathcal{NF}}$ KB is minimality-proper if it satisfies the following condition: for each $\mathbf{K}C \sqsubseteq D \in \Gamma$ and each $D(x) \in \mathcal{A}$

- (i) D does not contain a subconcept of the form $\forall \mathbf{A}R.E$; and
- (ii) D does not contain a subconcept of the form $\exists \mathbf{A}R.E$ in a disjunction.

For example, $\mathbf{K}C \sqsubseteq \mathbf{K}D \sqcup \exists \mathbf{A}R.C$ is not minimality-proper, while $\mathbf{K}C \sqsubseteq \exists \mathbf{A}R.C$ is.

The minimality-proper condition prevents $\forall \mathbf{A}R.C(x)$ to appear in \mathcal{B}' of the minimality condition 21. The fact the branch \mathcal{B}' is open implies that $(\mathcal{M}', \mathcal{M}'') \models \Sigma$, where $\mathcal{M}' = \{\mathcal{I} \mid \mathcal{I} \models Ob_{\mathbf{K}}(\mathcal{B}')$ and $\mathcal{M}'' = \{\mathcal{I} \mid \mathcal{I} \models Ob_{\mathbf{A}}(\mathcal{B}')$. The purpose of conditions (iii) and (iv) in Definition 21 is to imply that $\mathcal{M} = \{\mathcal{I} \mid \mathcal{I} \models Ob_{\mathbf{K}}(\mathcal{B})\}$ can replace \mathcal{M}'' to ensure that $(\mathcal{M}', \mathcal{M}) \models \Sigma$. This purpose would not be achieved if there were assertions of the form $\forall \mathbf{A}R.C(x) \in \mathcal{B}'$ because there could exist $R(x, y)$ such that $Ob_{\mathbf{K}}(\mathcal{B}) \models R(x, y)$ and $Ob_{\mathbf{A}}(f(\mathcal{B}')) \not\models R(x, y)$, which does not meet the requirement of “assuming as much as we know from \mathcal{B} ”. Because of the tableaux expansion rules, the branch \mathcal{B}' does not contain $\mathbf{A}R(x, y), C(y)$ and all the assertions obtained from $C(y)$. For analogous reason the minimality-proper KBs forbid $\exists \mathbf{A}R.E$ in disjunctions. We prove later that for the minimality-proper KBs, the minimality check is correct.

Definition 23 (Preferred Branch) A branch \mathcal{B} is preferred if all of the following conditions hold:

- (i) \mathcal{B} is pre-preferred;
- (ii) \mathcal{B} satisfies the minimality condition.

If there is a preferred branch \mathcal{B} for Σ , then there is an $\mathcal{ALCK}_{\mathcal{NF}}$ structure $(\mathcal{M}, \mathcal{M})$ that satisfies Σ and for each set of interpretations \mathcal{M}' , if $\mathcal{M}' \supset \mathcal{M}$ then $(\mathcal{M}', \mathcal{M})$ does not satisfy Σ . We conclude that \mathcal{B} is a model for Σ .

Lemma 24 (Termination) *Let Σ be a subjectively quantified and minimality-proper $\mathcal{ALCK}_{\mathcal{NF}}$ KB. Then the tableaux algorithm for checking satisfiability of Σ always terminates.*

To state completeness we must define what it means for a branch to represent a model. Let \mathcal{B} be a completed branch for Σ and \mathcal{M} a model for Σ . We define that \mathcal{B} represents \mathcal{M} if \mathcal{B} is preferred and there exists an injection $f : \mathcal{O}_{\mathcal{B}} \setminus \mathcal{O}_{\Sigma} \rightarrow \Delta \setminus \mathcal{O}_{\Sigma}$ such that $\mathcal{M} = \{\mathcal{I} \mid \mathcal{I} \models \text{Ob}_{\mathbf{K}}(f(\mathcal{B}))\}$ holds.

Theorem 25 (Soundness and completeness) *Let Σ be a subjectively quantified and minimality-proper $\mathcal{ALCK}_{\mathcal{NF}}$ KB. Then Σ is satisfiable if and only if there exists a preferred branch \mathcal{B} of the tableaux for Σ .*

If \mathcal{B} is a preferred branch for Σ , then $\mathcal{M} = \{\mathcal{I} \mid \mathcal{I} \models \text{Ob}_{\mathbf{K}}(\mathcal{B})\}$ is a model for Σ . If \mathcal{M} is a model for Σ , then there exists a completed branch \mathcal{B} for Σ that represents \mathcal{M} .

Theorem 26 (Decidability) *Let Σ be a subjectively quantified and minimality-proper $\mathcal{ALCK}_{\mathcal{NF}}$ KB. Then the satisfiability problem of Σ is decidable.*

4 Conclusion

For reasoning in $\mathcal{ALCK}_{\mathcal{NF}}$ we presented a tableaux algorithm with blocking. The blocking is employed to deal with modal part of MKNF-DL theory. This technique allows for reasoning about KBs which are both subjectively quantified and minimality-proper. This is a larger class of KBs than in previous approaches, which further restricted to simple KBs.

References

1. Donini, F. M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. *ACM Trans. Comput. Logic*, 3(2):177–225, 2002.
2. Gottlob, G.: Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, (2):397–425, 1992.
3. Ke, P., Sattler, U.: Next steps for description logics of minimal knowledge and negation as failure. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Description Logics*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
4. Malenko, J.: Reasoning in description logics. Charles University in Prague, dissertation thesis, 2012.
5. Marek, V. W., Truszczyński, M.: Nonmonotonic logic: context-dependent reasoning. Springer-Verlag, Berlin, 1993.
6. Niemela, I.: Autoepistemic logic as a unified basis for nonmonotonic reasoning. 1993.

A Protégé Plug-in for Defeasible Reasoning

Kody Moodley, Thomas Meyer, and Ivan Varzinczak

Centre for Artificial Intelligence Research
CSIR Meraka and University of KwaZulu-Natal, South Africa.
{kmoodley, tmeyer, ivarzinczak}@csir.co.za

Abstract. We discuss two approaches for *defeasible* reasoning in Description Logics that allow for the statement of defeasible subsumptions of the form “ α subsumed by β usually holds”. These approaches are known as *prototypical* reasoning and *presumptive* reasoning and are both rooted in the notion of *Rational Closure* developed by Lehmann and Magidor for the propositional case. Here we recast their definitions in a defeasible DL context and define algorithms for prototypical and presumptive reasoning in defeasible DL knowledge bases. In particular, we present a plug-in for the Protégé ontology editor which implements these algorithms for OWL ontologies. The plug-in is called *RaMP* and allows the modeller to indicate defeasible information in OWL ontologies and check entailment of defeasible subsumptions from defeasible knowledge bases.

1 Introduction

Entailment in standard DL reasoning systems is *monotonic*. Monotonicity is a property of knowledge representation (KR) systems that are built upon classical logics. It specifies that knowledge is always ‘incremental’. That is, adding to (or strengthening) the information in a Knowledge Base (KB) cannot result in any previously known conclusions being retracted from the KB. In classical logics, monotonic behaviour is exhibited on two levels. Firstly, on the meta-level where if a statement φ follows logically from a KB \mathcal{K} then φ also follows from any superset of \mathcal{K} and, secondly, on the object level from $\alpha \sqsubseteq \beta$ it follows that $\alpha \sqcap \gamma \sqsubseteq \beta$ for any γ .

It turns out that there are applications in which monotonicity is undesirable, i.e., *non-monotonic* reasoning is required. A typical scenario is when one needs to model *exceptions* in a domain. Let us consider an example. We select a domain which describes power plants (specifically *nuclear* power plants) and their safety under certain conditions. The following concept and role names will be used: PP (power plants), NPP (nuclear power plants), BrNPP (Brazilian nuclear power plants), SeismicArea (tracts of land prone to seismic activity), DangerousPP (power plants which are unsafe due to some conditions) and isLocln (a property of an entity indicating where it is located geographically). We specify a simple example using the vocabulary described above. Consider the following DL knowledge base

$$\mathcal{K} = \{ \text{BrNPP} \sqsubseteq \text{NPP}, \text{NPP} \sqsubseteq \neg \text{DangerousPP} \}$$

From \mathcal{K} we conclude classically that a Brazilian nuclear power plant is *not* a dangerous power plant ($\text{BrNPP} \sqsubseteq \neg\text{DangerousPP}$). This conclusion displays typical monotonic reasoning behaviour. That is, enforcing that *all* Brazilian nuclear power plants are nuclear power plants ($\text{BrNPP} \sqsubseteq \text{NPP}$) and that *all* nuclear power plants are *not* dangerous power plants ($\text{NPP} \sqsubseteq \neg\text{DangerousPP}$), it is also implicitly enforced that *all* Brazilian nuclear power plants are *not* dangerous power plants. \square

From an ontology modeller’s perspective, the type of reasoning behaviour exhibited in the example may not be suitable always. This is because the modeller may want to be able to cater for exceptions. We may not want to enforce that *all* nuclear power plants are safe. Rather, we may want to represent something less rigid. For example, one may want to express that in the most normal cases nuclear power plants are not dangerous but that there may be some exceptional cases in which they *are* dangerous (*defeasible* information). The broad approach to reasoning with KBs that contain defeasible information is known as *defeasible reasoning* and is a popular way to introduce non-monotonic reasoning behaviour into knowledge representation systems.

Returning to the example above, we argue that one needs to develop a defeasible reasoning approach to capture intuitions like: In general, nuclear power plants are not dangerous *but* a specific type of nuclear power plant (e.g., a Brazilian nuclear power plant located in a seismic area) may be dangerous. It would be useful to have a robust system in the DL setting that is capable of implementing this kind of reasoning.

The goal of this paper is to present and demonstrate a preliminary version of the software defeasible reasoning system that we have developed to be used with OWL ontologies. This system, known as *RAMP*, has been packaged as a plug-in for the popular Protégé ontology editor. The plug-in is discussed in more detail in Section 4.

2 Background

There are various approaches for introducing non-monotonic reasoning capabilities in logic-based knowledge representation systems. Among these, the approach by Kraus, Lehmann and Magidor (often called the KLM approach) [7,9] has been particularly successful due to its elegance and robustness. They enrich propositional logic with a defeasible ‘implication’ operator (\sim). This operator allows one to write down *defeasible implication* statements (also called *conditional assertions*) of the form $\alpha \sim \beta$, where α and β are propositional formulas. The sentence $\alpha \sim \beta$ intuitively means that in those situations where α is typically true, β is also true (for the precise semantics the reader should consult the provided references). Many extensions of the KLM approach have been proposed in the literature recently [10,6,2,5,3,4]. The approach that we use in our system is based on the KLM approach as well. Given a defeasible logic, such as the KLM-extension of propositional logic or the aforementioned extensions thereof and a *conditional KB* (a set of conditional assertions) it remains to be defined what a valid inference from a conditional KB is.

Lehmann et al. characterize the notion of *rational closure* [9, Section 5] and motivate this notion to be a suitable answer to the above question. The name given to the

reasoning approach that computes rational closure for conditional KBs is *prototypical reasoning*. Lehmann and colleagues also discuss similar forms of reasoning that may be suitable for computing inferences from conditional KBs. One such type of reasoning is known as *presumptive reasoning* [8]. Presumptive reasoning is actually a venturous extension of prototypical reasoning. This means that any inference that follows from a conditional KB using prototypical reasoning will also follow from the KB using presumptive reasoning. The converse is not necessarily true.

Our approach to defeasible reasoning is based on the work by Britz et al. [4] in which they propose a defeasible extension of the DL \mathcal{ALC} . The defeasibility is introduced through a *defeasible subsumption* operator (\sqsupseteq). This operator is ‘supraclassical’ to the classical subsumption operator (\sqsubseteq). The semantics of defeasible subsumption is similar to that for the \vdash operator given by Lehmann and colleagues. Intuitively the semantics states that given a defeasible subsumption axiom $\alpha \sqsupseteq \beta$ (where α and β may be complex \mathcal{ALC} concepts), then this statement means that the most *typical* α ’s are also β ’s (as opposed to *all* α ’s being β ’s in the classical case). As of writing, only a semantics for defeasible *subsumption* is explicitly provided by the approach (although defeasible *equivalence* follows trivially as well). Britz et al.’s approach is only applicable to DL TBoxes currently.

3 Prototypical and Presumptive Reasoning

Prototypical and presumptive reasoning are answers to the important question: given a KB that contains defeasible subsumption statements like $\alpha \sqsupseteq \beta$, what does it mean for one to draw inferences from this KB and how do we compute these inferences? Lehmann et al. have provided answers by developing the notions of prototypical and presumptive reasoning (albeit in a propositional context) and also specifying algorithms to compute these. It turns out that these notions are adaptable to the DL setting [4] and to this end we have adapted the algorithms by Lehmann et al. [9] for computing both prototypical and presumptive reasoning for a *defeasible KB* (an \mathcal{ALC} KB that may additionally contain defeasible subsumption statements of the form $\alpha \sqsupseteq \beta$).

3.1 Prototypical Reasoning

Prototypical reasoning corresponds exactly to the propositional notion of rational closure, lifted to the DL case. Due to space constraints, we are not concerned with the semantics here. Interested readers may consult the work of Lehmann and colleagues [9] for the semantics for rational closure in a propositional context. For a characterization of rational closure for the DL \mathcal{ALC} one can consult the work of Britz et al. [4].

The algorithm for prototypical reasoning, takes as input a subsumption statement (also called a *query*) which may be defeasible or classical and a defeasible KB. The output of the algorithm is **true** if the query φ follows (prototypically) from the given KB \mathcal{K} (denoted $\mathcal{K} \models_{Prot} \varphi$). Queries and defeasible KBs are currently restricted to subsumption statements for simplicity seeing that (in most cases) classical \mathcal{ALC} TBox axioms can be rewritten as classical \mathcal{ALC} subsumption statements.

The algorithm begins by performing a *classical transformation* of the input KB. Essentially this amounts to rewriting all defeasible statements in the KB as their classical counterparts and all classical statements into a specific normal form. The reason behind this transformation is two-fold: (i) It allows classical reasoning techniques to be used on the transformed KB and (ii) The normal form of the classical statements differentiates them (in the eyes of the algorithm) from the defeasible statements in the KB. As we shall see later, this last point also makes the next sub-procedure of the algorithm possible (the computation of a ranking of the statements in the KB). The transformation procedure is given below:

Definition 1 (transformKB). Given a defeasible KB \mathcal{K} :

$$\text{transformKB}(\mathcal{K}) := \{\text{transform}(\varphi) \mid \varphi \in \mathcal{K}\},$$

$$\text{where } \text{transform}(\varphi) := \begin{cases} \alpha \sqcap \neg\beta \sqsubseteq \perp, & \text{if } \varphi = \alpha \sqsubseteq \beta \\ \alpha \sqsubseteq \beta, & \text{if } \varphi = \alpha \lesssim \beta \end{cases}$$

Example 1. Let \mathcal{K} be the input of the procedure in Definition 1:

$$\mathcal{K} = \left\{ \begin{array}{l} \text{BrNPP} \sqsubseteq \exists\text{hasFault}, \text{BrNPP} \sqsubseteq \text{NPP}, \\ \text{BrNPP} \sqsubseteq \exists\text{isLoIn.Brazil}, \text{NPP} \lesssim \neg\exists\text{hasFault}, \\ \text{NPP} \sqcap \exists\text{isLoIn.SeismicArea} \lesssim \text{DangerousPP}, \\ \exists\text{hasFault} \lesssim \text{DangerousPP}, \text{DangerousPP} \lesssim \exists\text{isLoIn.SeismicArea} \end{array} \right\}$$

If we execute the procedure in Definition 1 for \mathcal{K} we get \mathcal{K}' :

$$\mathcal{K}' = \left\{ \begin{array}{l} \text{BrNPP} \sqcap \neg\exists\text{hasFault} \sqsubseteq \perp, \text{BrNPP} \sqcap \neg\text{NPP} \sqsubseteq \perp, \\ \text{BrNPP} \sqcap \neg\exists\text{isLoIn.Brazil} \sqsubseteq \perp, \text{NPP} \sqsubseteq \neg\exists\text{hasFault}, \\ \text{NPP} \sqcap \exists\text{isLoIn.SeismicArea} \sqsubseteq \text{DangerousPP}, \\ \exists\text{hasFault} \sqsubseteq \text{DangerousPP}, \text{DangerousPP} \sqsubseteq \exists\text{isLoIn.SeismicArea} \end{array} \right\}$$

□

The second procedure of the prototypical algorithm is the computation of a *ranking* of sentences in the KB according to the notion of *exceptionality* [9]. This procedure makes use of a sub-procedure *exceptional* which encodes the notion of exceptionality into the computation of the ranking. Before we specify the pseudocode for the complete procedure we define what we mean by the terms ranking and exceptionality.

Definition 2 (Ranking). Given a defeasible KB \mathcal{K} , a ranking for \mathcal{K} is a total preorder on the elements (axioms) in \mathcal{K} , with axioms higher up in the ordering interpreted as having a higher preference/importance.

Note that a ranking can in practice be implemented as a collection where each element of this collection is a set of sentences from the KB. Each sentence in a particular set has the same magnitude of importance (we also call this a *rank*). In the prototypical reasoning algorithm, the ranking of a defeasible KB is computed according to the

notion of exceptionality. Intuitively, a concept α is said to be exceptional w.r.t. a defeasible KB \mathcal{K} , if it is the case that $\top \sqsubseteq \neg\alpha$ usually follows from \mathcal{K} . That is, typically everything is in $\neg\alpha$, thereby making α an exception to this rule. In the context of our algorithm, checking whether α is exceptional w.r.t \mathcal{K} can be reduced to checking if \mathcal{K}' classically entails $\alpha \sqsubseteq \perp$ where \mathcal{K}' is the classical transformation of \mathcal{K} .

Definition 3 (Exceptionality). Let \mathcal{K} be a defeasible KB and α, β concepts. Then:

- α is exceptional w.r.t. \mathcal{K} iff $\mathcal{K}' \models \alpha \sqsubseteq \perp$. Where \mathcal{K}' is the transformation of \mathcal{K} .
- $\alpha \sqsubseteq \beta$ is exceptional w.r.t. \mathcal{K} iff α is exceptional w.r.t. \mathcal{K} .
- $\alpha \sqsubseteq \beta$ is exceptional w.r.t. \mathcal{K} iff $\alpha \sqcap \neg\beta$ is exceptional w.r.t. \mathcal{K} .
- $\mathcal{K}' \sqsubseteq \mathcal{K}$, is exceptional w.r.t. \mathcal{K} iff every element of \mathcal{K}' and only the elements of \mathcal{K}' are exceptional w.r.t. \mathcal{K} (we say that \mathcal{K}' is more exceptional than \mathcal{K}).

The above notion of exceptionality is included in the computation of the ranking for a defeasible KB as sub-procedure $exceptional(\mathcal{E}) := \{\alpha \sqsubseteq \beta \in \mathcal{E} \mid \mathcal{E} \models \alpha \sqsubseteq \perp\}$.

Now that we have detailed the sub-procedures and principles needed to describe the computation of a ranking of the sentences in a defeasible KB, we specify the complete procedure for implementing this:

Procedure $computeRanking(\mathcal{K})$

Input: Defeasible knowledge base \mathcal{K}

Output: The ranking, \mathcal{D} , for \mathcal{K}

```

1  $i := 0; \mathcal{K}' := transformKB(\mathcal{K}); \mathcal{E}_0 := \mathcal{K}'; \mathcal{E}_1 := exceptional(\mathcal{E}_0);$ 
2 while  $\mathcal{E}_{i+1} \neq \mathcal{E}_i$  do
3    $i := i + 1; \mathcal{E}_{i+1} := exceptional(\mathcal{E}_i);$ 
4  $n := i; \mathcal{D}_\infty := \mathcal{E}_n; \mathcal{D} := \{\mathcal{D}_\infty\};$ 
5 for  $j := 1$  to  $n$  do
6    $\mathcal{D}_j := \mathcal{E}_{j-1} \setminus \mathcal{E}_j; \mathcal{D} := \mathcal{D} \cup \{\mathcal{D}_j\};$ 
7 return  $\mathcal{D};$ 

```

Recall that the ranking of the KB is computed as a collection of sets of sentences (\mathcal{D} in Procedure $computeRanking$). Each element in any of such sets shares the same level of importance in the algorithm.

Example 2. Consider the transformed KB from Example 1:

$$\mathcal{K}' = \left\{ \begin{array}{l} \text{BrNPP} \sqcap \neg\exists\text{hasFault} \sqsubseteq \perp, \text{BrNPP} \sqcap \neg\text{NPP} \sqsubseteq \perp, \\ \text{BrNPP} \sqcap \neg\exists\text{isLoIn.Brazil} \sqsubseteq \perp, \text{NPP} \sqsubseteq \neg\exists\text{hasFault}, \\ \text{NPP} \sqcap \exists\text{isLoIn.SeismicArea} \sqsubseteq \text{DangerousPP}, \\ \exists\text{hasFault} \sqsubseteq \text{DangerousPP}, \text{DangerousPP} \sqsubseteq \exists\text{isLoIn.SeismicArea} \end{array} \right\}$$

\mathcal{K}' (second statement on Line 1 of the $computeRanking$ procedure) is calculated as in Example 1. Lines 2 to 3 of the $computeRanking$ procedure are responsible for computing the different levels of exceptional sentences according to Definition 3. This

segment of the procedure executes Procedure *exceptional* until $\mathcal{E}_{i+1} = \mathcal{E}_i$ (a fixed point is reached). In the example when Line 1 of *computeRanking* is executed $\mathcal{E}_0 := \mathcal{K}'$ and $\mathcal{E}_1 := \text{exceptional}(\mathcal{E}_0)$. If we do the computation as detailed in *exceptional* then we find: $\mathcal{E}_1 = \{\text{BrNPP} \sqcap \neg\exists\text{hasFault} \sqsubseteq \perp, \text{BrNPP} \sqcap \neg\text{NPP} \sqsubseteq \perp, \text{BrNPP} \sqcap \neg\exists\text{isLoIn.Brazil} \sqsubseteq \perp\}$.

We see that $\mathcal{E}_1 \neq \mathcal{E}_0$ therefore we have to execute the while loop again to find \mathcal{E}_2 . The result of this computation shows us that $\mathcal{E}_2 = \mathcal{E}_1$ and we have thus reached a fixed point. The while loop terminates and the final ranking \mathcal{D} can be computed as specified by Lines 4 to 6 of the procedure. The final ranking \mathcal{D} is composed of:

$$\mathcal{D}_\infty = \left\{ \begin{array}{l} \text{BrNPP} \sqcap \neg\exists\text{hasFault} \sqsubseteq \perp, \text{BrNPP} \sqcap \neg\text{NPP} \sqsubseteq \perp, \\ \text{BrNPP} \sqcap \neg\exists\text{isLoIn.Brazil} \sqsubseteq \perp \end{array} \right\}$$

$$\mathcal{D}_1 = \left\{ \begin{array}{l} \text{NPP} \sqsubseteq \neg\exists\text{hasFault}, \exists\text{hasFault} \sqsubseteq \text{DangerousPP}, \\ \text{NPP} \sqcap \exists\text{isLoIn.SeismicArea} \sqsubseteq \text{DangerousPP}, \\ \text{DangerousPP} \sqsubseteq \exists\text{isLoIn.SeismicArea} \end{array} \right\}$$

□

We now have a ranking for our original defeasible KB \mathcal{K} (see Example 1). Once this ranking is identified, then given a query, the core prototypical reasoning algorithm (see Algorithm 1) can be executed to determine if this query follows from the original defeasible KB. \mathcal{D}_∞ represents the infinite rank which contains the classical (non-defeasible) statements from the KB.

Algorithm 1: *Prototypical reasoning*

Input: The ranking \mathcal{D} for some KB, \mathcal{K} and a query φ of the form $\alpha \lesssim \beta$ (or $\alpha \sqsubseteq \beta$)

Output: **true** if $\mathcal{K} \models_{Prot} \alpha \lesssim \beta$ (or $\mathcal{K} \models_{Prot} \alpha \sqsubseteq \beta$), **false** otherwise

```

1  $n := 1$ ;
2 if  $\varphi = \alpha \sqsubseteq \beta$  then
3   return  $\mathcal{D}_\infty \models \alpha \sqsubseteq \beta$ ;
4 else if  $\varphi = \alpha \lesssim \beta$  then
5   while  $\bigcup \mathcal{D} \models \alpha \sqsubseteq \perp$  and  $\mathcal{D} \neq \emptyset$  do
6      $\mathcal{D} := \mathcal{D} \setminus \{\mathcal{D}_n\}; n := n + 1$ ;
7   return  $\bigcup \mathcal{D} \models \alpha \sqsubseteq \beta$ ;

```

We give an example to illustrate the prototypical reasoning algorithm below:

Example 3. Consider the following defeasible KB \mathcal{K} with six axioms:

$$\mathcal{K} = \left\{ \begin{array}{l} \text{BrNPP} \sqsubseteq \text{NPP}, \text{NPP} \sqsubseteq \text{PP}, \exists\text{isLoIn} \sqsubseteq \text{PP}, \\ \exists\text{isLoIn.SeismicArea} \sqsubseteq \text{DangerousPP}, \text{NPP} \lesssim \neg\text{DangerousPP}, \\ \text{BrNPP} \lesssim \exists\text{isLoIn.SeismicArea} \end{array} \right\}$$

Considering higher indices to represent more exceptional sentences, the ranking for this KB (computed using relevant procedures discussed) is the following:

$$\mathcal{D}_\infty = \left\{ \begin{array}{l} \text{BrNPP} \sqcap \neg\text{NPP} \sqsubseteq \perp, \exists\text{isLocln} \sqcap \neg\text{PP} \sqsubseteq \perp, \\ \exists\text{isLocln.SeismicArea} \sqcap \neg\text{DangerousPP} \sqsubseteq \perp, \text{NPP} \sqcap \neg\text{PP} \sqsubseteq \perp \end{array} \right\}$$

$$\mathcal{D}_1 = \{\text{BrNPP} \sqsubseteq \exists\text{isLocln.SeismicArea}\}$$

$$\mathcal{D}_2 = \{\text{NPP} \sqsubseteq \neg\text{DangerousPP}\}$$

If we compute prototypical reasoning (via Algorithm 1) for the following query $\varphi = \text{BrNPP} \sqsubseteq \text{DangerousPP}$ then we get the positive result $\mathcal{K} \models_{Prot} \varphi$. We find the following motivation for this: the condition on Line 5 of the algorithm holds because $\text{BrNPP} \sqsubseteq \perp$ follows from \mathcal{D} ($\mathcal{D}_\infty \cup \mathcal{D}_1 \cup \mathcal{D}_2$) and $\mathcal{D} \neq \emptyset$. Therefore we execute the loop body and \mathcal{D} becomes $\mathcal{D}_\infty \cup \mathcal{D}_1$. The loop condition no longer holds because $\bigcup \mathcal{D} \not\models \text{BrNPP} \sqsubseteq \perp$ and therefore the loop terminates.

Last, the classical entailment check ($\bigcup \mathcal{D} \models \text{BrNPP} \sqsubseteq \text{DangerousPP}$) on Line 7 is performed. This returns **true** in our example and therefore $\mathcal{K} \models_{Prot} \varphi$.

An intuitive reading of this result shows us that *typically* nuclear power plants are *not* dangerous power plants ($\text{NPP} \sqsubseteq \neg\text{DangerousPP}$). If we examine the query φ we see that we are not in a typical situation because we have a *specific type* of nuclear power plant, i.e., A *Brazilian* nuclear power plant. Additionally, we know that typical Brazilian nuclear power plants are located in seismic areas stated by ($\text{BrNPP} \sqsubseteq \exists\text{isLocln.SeismicArea}$) and typical power plants located in seismic areas are dangerous ($\exists\text{isLocln} \sqsubseteq \text{PP}$, $\exists\text{isLocln.SeismicArea} \sqsubseteq \text{DangerousPP}$). Given this information, prototypical reasoning allows for the possibility that this type of nuclear power plant *is* dangerous, which represents an exception to the general rule that $\text{NPP} \sqsubseteq \neg\text{DangerousPP}$. \square

Next we introduce an alternative defeasible reasoning algorithm to the prototypical approach just described. This approach is known as presumptive reasoning.

3.2 Presumptive Reasoning

The presumptive reasoning algorithm is a venturesome extension of the prototypical one. Essentially the difference between them (from an algorithmic perspective) is that presumptive reasoning computes an extended version of the ranking that prototypical reasoning does. Presumptive reasoning has a semantics for the propositional case [8] which we shall not discuss here due to space constraints. The algorithm for computing presumptive reasoning is more lenient than prototypical reasoning in allowing sentences to follow from a defeasible KB, i.e., it *presumes* that some sentence follows from the KB as long as there is no evidence it can find to the contrary. Both algorithms are virtually the same barring one difference: the ranking of sentences in the input KB is computed slightly differently in presumptive reasoning. Because a presumptive ranking is an extension of a prototypical ranking, we describe a procedure for converting a prototypical ranking for a defeasible KB into a presumptive one.

We let \mathcal{D} be a prototypical ranking for some defeasible KB \mathcal{K} . Each element $\mathcal{D}_i \in \mathcal{D}$ (which we refer to as a *rank*) is a set of sentences from \mathcal{K} . To convert \mathcal{D} to a presumptive ranking \mathcal{D}' we add $|\mathcal{D}_i| - 1$ ranks *above* each \mathcal{D}_i in \mathcal{D} .

Example 4. Let $\mathcal{D}_i = \{\varphi_1, \varphi_2, \varphi_3\}$ be a rank in some prototypical ranking \mathcal{D} : We recall that in order to extend \mathcal{D} to a presumptive ranking we need to add $|\mathcal{D}_i| - 1$ ranks above each \mathcal{D}_i in \mathcal{D} . For our example, let us consider that we have a prototypical ranking containing just one rank \mathcal{D}_i as defined above. We thus need to add two ranks above \mathcal{D}_i in \mathcal{D} . To understand what these two ‘presumptive’ ranks will look like we have to explain how a prototypical ranking is used in the prototypical algorithm.

Recall that Lines 5-6 of Algorithm 1 essentially finds a maximal subset of the axioms in \mathcal{D} in which the antecedent of the query is satisfiable. However, the prototypical algorithm does this in a ‘clunky’ way by removing *entire* ranks at a time (Line 6). Presumptive reasoning is more thorough in finding a maximal subset of the axioms in \mathcal{D} because it first tries removing (all permutations of) *one* axiom from the highest rank, instead of the entire rank. If this does not make the antecedent satisfiable then it tries removing (all permutations of) *two* axioms from the same rank etc. until eventually (all permutations of) $|\mathcal{D}_i| - 1$ axioms are removed. We now show an elegant way to perform this fine-grained removal.

We start with the initial rank $\mathcal{D}_i = \{\varphi_1, \varphi_2, \varphi_3\}$. We want to only remove \mathcal{D}_i from \mathcal{D} as a last resort but before that we want to try removing all permutations of one axiom and then two. Instead of doing this in a naïve way by computing all these permutations, we use the fact that *removing* one axiom is the same as *keeping* two in a set of three axioms as in the example. In other words we would like to compute all instances of *two* axioms holding simultaneously in our \mathcal{D}_i . That is $(\varphi_1 \mathbf{and} \varphi_2) \mathbf{or} (\varphi_1 \mathbf{and} \varphi_3) \mathbf{or} (\varphi_2 \mathbf{and} \varphi_3)$ in our example.

It turns out that there is a way to transform this ‘check’ or expression into a *single axiom* according to the following rules for DL axioms:

$$\begin{aligned} \mathbf{and}(\alpha_i \sqsubseteq \beta_i)_{i=1}^n &= \top \sqsubseteq \prod_{i=1}^n (\neg \alpha_i \sqcup \beta_i) \mathbf{and}, \\ \mathbf{or}(\alpha_i \sqsubseteq \beta_i)_{i=1}^n &= \prod_{i=1}^n \alpha_i \sqsubseteq \sqcup_{i=1}^n \beta_i, \text{ where } \alpha_i \sqsubseteq \beta_i = \varphi_i. \end{aligned}$$

If removing all permutations of one axiom (keeping two) does not make the antecedent satisfiable in our example then we try removing all permutations of *two* (keeping one). Therefore for our example the expression to check to verify this is $\varphi_1 \mathbf{or} \varphi_2 \mathbf{or} \varphi_3$. Therefore the presumptive conversion \mathcal{D}'_i for the prototypical rank \mathcal{D}_i is $\{((\varphi_1 \mathbf{and} \varphi_2) \mathbf{or} (\varphi_1 \mathbf{and} \varphi_3) \mathbf{or} (\varphi_2 \mathbf{and} \varphi_3)), (\varphi_1 \mathbf{or} \varphi_2 \mathbf{or} \varphi_3), \{\varphi_1, \varphi_2, \varphi_3\}\}$. This process is repeated on each \mathcal{D}_i in a prototypical ranking and the final presumptive ranking \mathcal{D}' is computed as $\bigcup \mathcal{D}'_i$. \square

It is important to note that other than the difference in the computation of the ranking, the prototypical and presumptive reasoning algorithms are identical. Therefore Algorithm 1 can be executed with a presumptive ranking as input to compute presumptive reasoning for a given defeasible KB and query. We have developed a Protégé plug-in that implements both prototypical and presumptive reasoning for DL-based (and therefore OWL) ontologies. We present this plug-in in the next section.

4 RaMP

We have developed a plug-in for Protégé which allows the ontology modeller to represent defeasible information in the ontology without explicitly extending the underlying ontology language (OWL). This is possible through axiom annotations in Protégé which do not affect the logical meaning of the ontology. Furthermore, this plug-in implements the prototypical and presumptive reasoning algorithms. The plug-in is called *RaMP*¹ which stands for Rational Monotonicity Plug-in. Figure 1 depicts the control panel of RaMP's interface in Protégé 4.1.

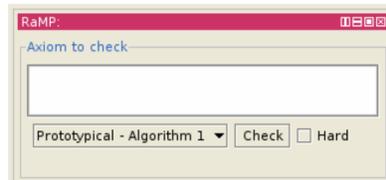


Fig. 1: RaMP: Reasoning controller panel

This component of the RaMP interface is called the reasoning controller panel. This panel provides a text input field for the user to enter an axiom (defeasible or hard). The axiom can be verified to follow (or not follow) from the ontology, based on the currently selected reasoning algorithm. This task is accomplished by clicking the “check” button in the same panel. The reasoning algorithm can be selected from a drop-down menu. Currently, only prototypical and presumptive reasoning algorithms are available. There is also a checkbox for indicating if the axiom entered is defeasible or classical. If the axiom is classical, we also refer to it as a *hard* axiom. For convenience, RaMP provides a window display (Figure 2) in Protégé to show the user which axioms in the loaded ontology are indicated as defeasible axioms.

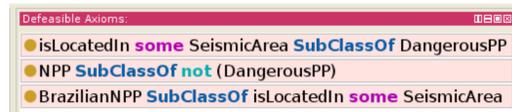


Fig. 2: RaMP: Defeasible axioms display

This brings us to the topic of how to assert that an axiom is defeasible in the loaded ontology. RaMP indicates axioms as defeasible in the ontology through a “flagging” process. In the Class Description window in Protégé, the superclasses and equivalent classes of the selected class are indicated. Next to each of these classes there is a button, labelled “d” for defeasibility, which can be clicked to toggle whether that axiom should be viewed as defeasible or not by RaMP. When an axiom is indicated as defeasible the button turns pink and an axiom *annotation* is added to the ontology which stores this information (see Figure 3).

¹ <http://code.google.com/p/nomor/>



Fig. 3: RaMP: Toggling defeasibility of axioms

The reasoning algorithms implemented by RaMP work in a similar way. They compute a ranking (ordering) of the axioms in the ontology and use this ranking to determine which axioms are more important than others in the ontology. The details of these algorithms will be provided in Sections 3.1 and 3.2 respectively. An example ranking is depicted below in Figure 4. The axioms appearing on the light pink background are defeasible axioms while the others are hard.



Fig. 4: RaMP: Axiom ranking display

As an added feature of RaMP, we have also included a rudimentary facility for the user to fine-tune the computed ranking for the ontology. Adjacent to the defeasible toggle switch in Protégé, there is a button labelled “r” to prompt the user to enter a numerical value representing the rank of the selected axiom. The specific numerical value chosen does not matter. What matters is the relative ordering between these values. If axiom φ has rank 1 and it is necessary for axiom φ' to have a higher rank then it does not matter whether axiom φ has rank 2 or 200 as long as the rank of φ' is greater than 2 or 200 respectively. The axiom ranking feature is depicted in Figure 5.



Fig. 5: RaMP: Axiom ranking feature

5 Conclusion

We have presented a description of a Protégé plug-in which implements a preliminary version of non-monotonic reasoning for DL-based ontologies. The plug-in provides a mechanism for indicating defeasible information in the ontology as well as an implementation of two defeasible reasoning algorithms adapted from the work of Lehmann and colleagues [9] in a propositional setting. The plug-in is still in the early stages of development. We would like to introduce a more elaborate interface to facilitate better integration with Protégé, we are also investigating potential optimizations for the defeasible reasoning algorithms. In the future we also plan to include: (i) ABox reasoning; (ii) Defeasibility in other OWL constructs such as disjointness, roles and role properties; (iii) More sophisticated reasoning tasks available in standard monotonic systems such as Classification, Instance checking etc [1] and (iv) Other versions of defeasible reasoning (in addition to the ones discussed here). Currently we are evaluating the results reported by the algorithms implemented in the tool. The results will be evaluated in various application domains to identify where each algorithm is most suitable as a reasoning tool.

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF. The work of Ivan Varzinczak was also supported by the National Research Foundation under Grant number 81225.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The description logic handbook. Cambridge, 2 edn. (2007)
2. Britz, K., Heidema, J., Meyer, T.: Semantic preferential subsumption. In: Proc. of KR. pp. 476–484 (2008)
3. Britz, K., Meyer, T., Varzinczak, I.: Preferential reasoning for modal logics. Electronic Notes in Theoretical Computer Science 278, 55–69 (2011), proc. of the Workshop on Methods for Modalities
4. Britz, K., Meyer, T., Varzinczak, I.: Semantic foundation for preferential description logics. In: Proc. of the Australasian Conference on Artificial Intelligence (2011)
5. Casini, G., Straccia, U.: Rational closure for defeasible description logics. In: Proc. of JELIA. pp. 77–90 (2010)
6. Giordano, L., Olivetti, N., Gliozzi, V., Pozzato, G.: $\mathcal{ALC} + T$: A preferential extension of description logics. Fundamenta Informaticae 96(3), 341–372 (2009)
7. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44, 167–207 (1990)
8. Lehmann, D.: Another perspective on default reasoning. Annals of Mathematics and Artificial Intelligence 15, 61–82 (1995)
9. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55(1), 1–60 (1992)
10. Quantz, J.: A preference semantics for defaults in terminological logics. In: Proc. of KR. pp. 294–305 (1992)

A Decidable Extension of \mathcal{SRIQ} with Disjunctions in Complex Role Inclusion Axioms

Milenko Mosurović¹, Henson Graves², and Nenad Krdžavac³

¹ Faculty of Mathematics and Natural Science, University of Montenegro, Podgorica, ul. Džordža Vašingtona bb, 81000 Podgorica, Montenegro.

`milenko@ac.me`

² Algos Associates, 2829 West Cantey Street, Fort Worth, TX 76109 United States
`henson.graves@hotmail.com`

³ Department of Accounting and Information Systems, College of Business and Law, University College Cork, Cork City, Ireland.

`N.Krdzavac@ucc.ie`

Abstract. This paper establishes the decidability of $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ which has composition-based role Inclusion axioms (RIAs) of the form $R_1 \circ \dots \circ R_n \sqsubseteq T_1 \sqcup \dots \sqcup T_m$. Also the consistency of an Abox \mathcal{A} of $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ DL w.r.t. Rbox \mathcal{R} is established. Motivation for this kind of RIAs comes from applications in the field of manufactured products as well as other conceptual modeling applications such as family relationships. The solution is based on a tableau algorithm.

Keywords: Description Logic, Manufacturing system, Tableau, Composition-based Role Inclusion Axiom.

1 Introduction

Description logic (DL) [1] has focused on extending decidability results to DLs with more complex RIAs [6, 7, 9]. However, the logic \mathcal{SROIQ} DL which is logical basis for the standard Ontology Web Language OWL 2 [3], does not admit assertions which have role unions on the right hand side of RIAs. Many applications involve RIAs with role unions on the right side. For example in modeling an engine in a car that can power wheelInCar or oilPump or generator, or all of these, at the same time [8, 2]. This model can be described in the following composition-based RIAs [11]:

$$\text{engineInCar} \circ \text{powers} \sqsubseteq \text{wheelInCar} \sqcup \text{generatorInCar} \sqcup \text{oilPunInCar} \quad (1)$$

One can conclude that for an individual car c_1 and an individual p_1 : if p_1 is powered by an individual engine e_1 in the car c_1 then p_1 is an individual wheel or a generator or an oilpump in c_1 . The RIA of the form (1) can be expressed in an extension of \mathcal{ALC} DL with composition-based RIAs [11], but \mathcal{SROIQ} DL does not support such composition-based RIAs. Modeling such RIAs in the

extensions of \mathcal{ACL} DL considered only two roles on the left hand side of the RIAs. This paper introduces the $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ DL that extends \mathcal{SRIQ} DL [5] with composition-based RIAs of the form (2). As noted in [11] the RIA of the form (2) are not role value-maps [10]. The logic analyzed in this paper overcomes the following shortcomings of the logics studied in [11]:

1. Finite automata handle composition-based RIAs of the form (2).
2. Does not require a Rbox to be admissible [11],
3. Does not require all roles to be disjoint [11],
4. Allows more than two roles on the left hand side of composition-based RIAs.

The rest of the paper is organized as follows. Next section gives definition of $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ DL. Section 3 defines tableau for $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ and proves decidability of the logic. The section also gives an example of tableau for RIA of the form (2). The last section concludes the paper.

2 Preliminaries

The alphabet of \mathcal{SRIQ} and $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ DL consists of set of concept names \mathcal{N}_C , set of role names \mathcal{N}_R , set of simple role names $\mathcal{N}_S \subset \mathcal{N}_R$ and finally, a set of individual names \mathcal{N}_I . The set of roles is $\mathcal{N}_R \cup \{R^- \mid R \in \mathcal{N}_R\}$ and on this set the function $Inv(\cdot)$ is defined as $Inv(R) = R^-$ and $Inv(R^-) = R$ for $R \in \mathcal{N}_R$. A role chain is a sequence of roles $w = R_1 R_2 \dots R_n$.

$\mathcal{SR}^{\sqcup}\mathcal{IQ}$ language is an extension of \mathcal{SRIQ} [5], by allowing new kinds of RIAs in role hierarchy. The syntax of the $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ DL concepts, Rbox, Tbox and Abox are given in definitions 1, 2 and 3 following [5].

Definition 1. *Set of $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ concepts is a smallest set such that*

- every concept name and \top , \perp are concepts, and,
- if C and D are concept and R is a role, S is simple role, n is non-negative integer, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\forall R.C$, $\exists R.C$, $\exists S.Self$, $(\leq nS.C)$, $(\geq nS.C)$ are concepts.

A general concept inclusion axiom (GCI) is an expression of the form $C \sqsubseteq D$ for two $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ -concepts C and D . A Tbox \mathcal{T} is a finite set of GCIs.

An individual assertion has one of the following forms: $a : C$, $(a, b) : R$, $(a, b) : \neg S$, or $a \neq b$, for $a, b \in \mathcal{N}_I$ (the set of individual names), a (possibly inverse) role R , a (possibly inverse) simple role S , and a $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ -concept C . A $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ -Abox \mathcal{A} is a finite set of individual assertions.

A (composition-based) RIA is a statement of the form [11]:

$$R_1 \dots R_n \stackrel{\sqsubseteq}{\sqsubseteq} T_1 \sqcup \dots \sqcup T_m. \quad (2)$$

Without additional restrictions on RIAs, some DLs [11] with composition-based RIAs are undecidable.

Definition 2. Strict partial order \prec (irreflexive, transitive, and antisymmetric), on the set of roles, provides acyclicity [5]. Allowed RIAs in $SRIQ$ DL with respect to \prec , are expressions of the form $w \sqsubseteq R$, where [4, 5]:

1. R is a simple role name, $w = S$ is a simple role, and $S \prec R$ or $S = R^-$ or
2. $R \in \mathcal{N}_R \setminus \mathcal{N}_S$ is a role name and
 - $w = RR$, or
 - $w = R^-$, or
 - $w = S_1 \cdots S_n$ and $S_i \prec R$, for $1 \leq i \leq n$, or
 - $w = RS_1 \cdots S_n$ and $S_i \prec R$, for $1 \leq i \leq n$, or
 - $w = S_1 \cdots S_n R$ and $S_i \prec R$, for $1 \leq i \leq n$

A $SRIQ$ role hierarchy is a finite set \mathcal{R}_h^1 of RIAs. A $SRIQ$ role hierarchy \mathcal{R}_h^1 is regular if there exists strict partial order \prec such that each RIA in \mathcal{R}_h^1 is allowed with respect to \prec [4, 5].

Definition 3. A $SR^{\sqcup}IQ$ role hierarchy is a finite set $\mathcal{R}_h = \mathcal{R}_h^1 \cup \mathcal{R}_h^2$, where \mathcal{R}_h^1 is $SRIQ$ role hierarchy and \mathcal{R}_h^2 is set of RIA $R_{i1} \cdots R_{in_i} \sqsubseteq T_{i1} \sqcup \cdots \sqcup T_{im_i}$, and T_{ij} are not simple roles, for $i = 1, \dots, k$. A $SR^{\sqcup}IQ$ role hierarchy \mathcal{R}_h is regular if \mathcal{R}_h^1 is regular and T_{ij} does not appear on the left hand side of RIAs in \mathcal{R}_h . A $SR^{\sqcup}IQ$ set of role assertions is a finite set \mathcal{R}_a of the assertions $Ref(R)$, $Irr(S)$, $Sym(R)$, $Tra(V)$, and $Dis(T, S)$, where R is a role, S, T are simple roles and V is not simple role [5]. A $SR^{\sqcup}IQ$ Rbox $\mathcal{R} = \mathcal{R}_h \cup \mathcal{R}_a$, where \mathcal{R}_h is $SR^{\sqcup}IQ$ role hierarchy and \mathcal{R}_a is a set of role assertions.

If \mathcal{R}_h^1 is regular w.r.t strict partial order \prec then we extend \prec such that $R_{ij} \prec T_{il}$ hold, $i = 1, \dots, k$ and $j = 1, \dots, n_i$, $l = 1, \dots, m_i$. Further, we assume that labels, such as $k, n_i, m_i, T_{il}, R_{ij}$, have the same meaning as defined in definition 3.

Definition 4. The semantics of the $SR^{\sqcup}IQ$ DL is defined by using interpretation. An interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called the domain of the interpretation. A valuation $\cdot^{\mathcal{I}}$ associates: every concept name C with a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$; every role name R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and, every individual name a with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ [1].

Definition 5. An interpretation \mathcal{I} extends to $SR^{\sqcup}IQ$ complex concepts and roles according to the following semantic rules:

- If R is a role name, then $(R^-)^{\mathcal{I}} = \{\langle x, y \rangle : \langle y, x \rangle \in R^{\mathcal{I}}\}$,
- If R_1, R_2, \dots, R_n are roles then $(R_1 R_2 \dots R_n)^{\mathcal{I}} = (R_1)^{\mathcal{I}} \circ (R_2)^{\mathcal{I}} \circ \dots \circ (R_n)^{\mathcal{I}}$ and $(R_1 \sqcup R_2 \dots \sqcup R_n)^{\mathcal{I}} = (R_1)^{\mathcal{I}} \cup (R_2)^{\mathcal{I}} \cup \dots \cup (R_n)^{\mathcal{I}}$, where $sign \circ$ is a composition of binary relations,
- If C and D are concepts, R is a role, S is a simple role and n is a non-negative integer, then ⁴
 - $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}, \perp^{\mathcal{I}} = \emptyset, (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},$
 - $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, (\exists R.C)^{\mathcal{I}} = \{x : \exists y. \langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\},$

⁴ $\#M$ denotes cardinality of set M .

$$\begin{aligned}
(\exists S.Self)^{\mathcal{I}} &= \{x : \langle x, x \rangle \in S^{\mathcal{I}}\}, (\forall R.C)^{\mathcal{I}} = \{x : \forall y. \langle x, y \rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}, \\
(\geq nS.C)^{\mathcal{I}} &= \{x : \#\{y : \langle x, y \rangle \in S^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \geq n\}, \\
(\leq nS.C)^{\mathcal{I}} &= \{x : \#\{y : \langle x, y \rangle \in S^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \leq n\}.
\end{aligned}$$

Inference problems for $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ are defined in standard way [5].

Definition 6. An interpretation \mathcal{I} satisfies a RIA $R_1 \cdots R_n \sqsubseteq T_1 \sqcup \cdots \sqcup T_m$, if $R_1^{\mathcal{I}} \circ \cdots \circ R_n^{\mathcal{I}} \subseteq T_1^{\mathcal{I}} \cup \cdots \cup T_m^{\mathcal{I}}$. An interpretation \mathcal{I} is model of a

- Tbox \mathcal{T} (written $\mathcal{I} \models \mathcal{T}$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each GCI $C \sqsubseteq D$ in \mathcal{T} .
- role hierarchy \mathcal{R}_h , if it satisfies all RIAs in \mathcal{R}_h (written $\mathcal{I} \models \mathcal{R}_h$).
- role assertions \mathcal{R}_a (written as $\mathcal{I} \models \mathcal{R}_a$) if $\mathcal{I} \models \Phi$ holds for each role assertion axiom $\Phi \in \mathcal{R}_a$, where is $\mathcal{I} \models Dis(S, R)$ if $S^{\mathcal{I}} \cap R^{\mathcal{I}} = \emptyset$,
 $\mathcal{I} \models Sym(R)$ if $R^{\mathcal{I}}$ is symmetric relation, $\mathcal{I} \models Tra(R)$ if $R^{\mathcal{I}}$ is transitive relation,
 $\mathcal{I} \models Ref(R)$ if $R^{\mathcal{I}}$ is reflexive relation, $\mathcal{I} \models Irr(S)$ if $R^{\mathcal{I}}$ is irreflexive relation.
- Rbox $\mathcal{R} = \langle \mathcal{R}_h, \mathcal{R}_a \rangle$ (written as $\mathcal{I} \models \mathcal{R}$) if $\mathcal{I} \models \mathcal{R}_h$ and $\mathcal{I} \models \mathcal{R}_a$.
- Abox \mathcal{A} ($\mathcal{I} \models \mathcal{A}$) if for all individual assertions $\phi \in \mathcal{A}$ we have $\mathcal{I} \models \phi$, where
 $\mathcal{I} \models a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, $\mathcal{I} \models a \neq b$ if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$,
 $\mathcal{I} \models (a, b) : R$ if $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, $\mathcal{I} \models (a, b) : \neg R$ if $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \notin R^{\mathcal{I}}$.

For an interpretation \mathcal{I} , an element $x \in \Delta^{\mathcal{I}}$ is called an instance of a concept C if $x \in C^{\mathcal{I}}$. An Abox \mathcal{A} is consistent with respect to a Rbox \mathcal{R} and a Tbox \mathcal{T} if there is a model \mathcal{I} for \mathcal{R} and \mathcal{T} such that $\mathcal{I} \models \mathcal{A}$.

Definition 7. A concept C is called satisfiable if there is an interpretation \mathcal{I} with $C^{\mathcal{I}} \neq \emptyset$. A concept D subsumes a concept C (written $C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each interpretation. Two concepts are equivalent (written $C \equiv D$) if they are mutually subsuming.

All standard inference problems for $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ -concepts and Abox can be reduced [5] to the problem of determining the consistency of a $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ -Abox w.r.t. a Rbox, where we can assume w.l.o.g. that all role assertions in the Rbox are of the form $Dis(S, R)$. We call such Rbox reduced.

3 The Extension of \mathcal{SRIQ} Tableau

Let \mathcal{A} be a $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ -Abox and \mathcal{R} a reduced $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ -Rbox and let $\mathcal{R}_{\mathcal{A}}$ be a set of role names appearing in \mathcal{A} and \mathcal{R} , including their inverse, and $\mathcal{I}_{\mathcal{A}}$ is the set of individual names appearing in \mathcal{A} . To check whether Abox \mathcal{A} is consistent w.r.t. Rbox \mathcal{R} we transform $\mathcal{SR}^{\sqcup}\mathcal{IQ}$ -Rbox \mathcal{R} to \mathcal{SRIQ} -Rbox \mathcal{R}' as follows:

1. For each role name $R \in \mathcal{R}_{\mathcal{A}}$ we define equivalence class $[R] = \{R\}$ and set $[R^-] = [R]^-$, $comp([R]) = \{R\}$, $comp([R^-]) = \{R^-\}$,
2. For each RIA of the form $R_{i1} \cdots R_{in_i} \sqsubseteq T_{i1} \sqcup \cdots \sqcup T_{im_i} \in \mathcal{R}$ ($1 \leq i \leq k$) we define equivalence class $[T_{i1} \sqcup \cdots \sqcup T_{im_i}] = \{T_{j1} \sqcup \cdots \sqcup T_{jm_j} \mid \{T_{i1}, \dots, T_{im_i}\} = \{T_{j1}, \dots, T_{jm_j}\}, 1 \leq j \leq k\}$ and set $comp([T_{i1} \sqcup \cdots \sqcup T_{im_i}]) = \{T_{i1}, \dots, T_{im_i}\}$

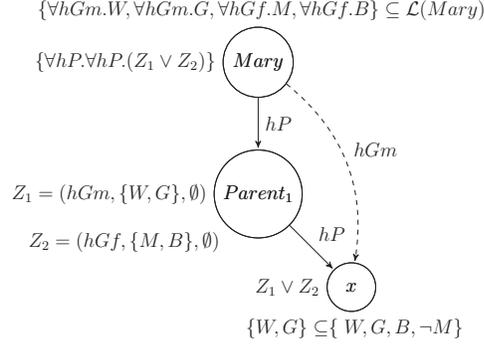


Fig. 1. A part of tableau for (3) and (4)

3. We consider equivalence classes $[R]$, previously defined, as role names which do not appear in \mathcal{R}_A . Set of the role names is denoted with \mathcal{R}'_A . Let's define $\mathcal{R}' = \{[R_1] \cdots [R_n] \sqsubseteq [T_1 \sqcup \cdots \sqcup T_m] \mid R_1 \cdots R_n \sqsubseteq T_1 \sqcup \cdots \sqcup T_m \in \mathcal{R}\}$.

If Rbox \mathcal{R} is regular w.r.t order \prec then Rbox \mathcal{R}' is regular w.r.t \prec' defined as follows $[R] \prec' [S]$ iff $R \prec S$ and $[T_{ij}] \prec' [T_{i1} \sqcup \cdots \sqcup T_{im_i}]$, $j = 1, \dots, m_i$, $i = 1, \dots, k$. Equivalence classes and order \prec' previously defined are using for automata construction. For the following example of RIAs $R_1R_2 \sqsubseteq H_1 \sqcup H_2^-$ and $S_1S_2 \sqsubseteq H_2^- \sqcup H_1$ one should construct a nondeterministic finite automaton (NFA) for role $[H_1 \sqcup H_2^-]$. The automaton should accept words R_1R_2 and S_1S_2 . Namely, for every role $[R]$ we have kept the construction of NFA $\mathcal{B}_{[R]}$ based on \mathcal{R}' , as same as defined in [5]. For \mathcal{B} an NFA and q a state of \mathcal{B} , \mathcal{B}^q denotes the NFA obtained from \mathcal{B} by making q the (only) initial state of \mathcal{B} [5]. The language recognized by NFA \mathcal{B} is denoted by $\mathcal{L}(\mathcal{B})$.

To illustrate main idea in this paper, we use the following simple example.

Example 1. In this example we use the following abbreviations: $hP = hasParent$, $hGm = hasGrandMother$, $hGf = hasGrandFather$, $W = Woman$, $M = Man$, $G = Gentle$, $B = Blabber$. We defined the following RIA:

$$hP \circ hP \sqsubseteq hGm \sqcup hGf \quad (3)$$

and the individual assertion:

$$Mary : \forall hGm.W \sqcap \forall hGf.M \sqcap \forall hGm.G \sqcap \forall hGf.B \quad (4)$$

We should decide whether x (see Fig. 1) is instance of *GrandMother* or *GrandFather*. If $x \in GrandMother^{\mathcal{I}}$ then $x \in W^{\mathcal{I}}$, $x \in G^{\mathcal{I}}$. In the case of $(Mary, x) \in hGm^{\mathcal{I}}$, it does not break syntax rules. Similar to this one, if $x \in GrandFather^{\mathcal{I}}$ then $x \in M^{\mathcal{I}}$, $x \in B^{\mathcal{I}}$ and $(Mary, x) \in hGf^{\mathcal{I}}$ hold. Meta-labels Z_1 and Z_2 are using to remember the (relevant) parts of the labels in the

node *Mary* which should be transferred from the node to node x (see Fig. 1). First component in Z_1 is role. The second component is the set of the concepts $\{C \mid \text{Mary is instance of concept } \forall hGm.C\}$. The third component is the set of concepts, for which *Mary* is instance and should be superset of the set $\{C \mid x \text{ is instance of concept } \forall hGm^-.C\}$. Because of inverse role we need first and third component. To choose given meta-label, we note as $Z_1 \vee Z_2$. To recognize path $hP \circ hP$ from node *Mary* to x we use NFA $\mathcal{B}_{[hGm \sqcup hGf]}$ noted as follows $\forall \mathcal{B}_{[hGm \sqcup hGf]}.(Z_1 \vee Z_2)$. \square

We assume that all concepts are in negation normal form (NNF). For given concept C_0 , $clos(C_0)$ is the smallest set that contains C_0 and that is closed under sub-concepts and $\dot{\cdot}$. We use $\dot{\cdot}C$ for NNF of $\neg C$ [5]. We use two sets of the label of nodes. First set is [5]: $clos(\mathcal{A}) := \cup_{a:C \in \mathcal{A}} clos(C)$. The second set is: $NFAclos(\mathcal{A}, \mathcal{R}) := \{\forall \mathcal{B}_{[R]}^q.Z \mid [R] \in \mathcal{R}'_{\mathcal{A}} \text{ and } q \text{ is state in NFA } \mathcal{B}_{[R]} \text{ and } Z = \bigvee_{T \in comp([R])} (T, Z_T, \hat{Z}_T), Z_T \subseteq clos(\mathcal{A})|_T, \hat{Z}_T \subseteq clos(\mathcal{A})|_{T^-}\}$, where $clos(\mathcal{A})|_Q = \{C \mid \forall Q.C \in clos(\mathcal{A})\}$.

In the proofs of decidability we use set $PL(\mathcal{B}_{[R]}) = \{\langle w', q \rangle \mid q \text{ is a state in } \mathcal{B}_{[R]}, (\forall w'' \in \mathcal{L}(\mathcal{B}_{[R]}^q))(w'w'' \in \mathcal{L}(\mathcal{B}_{[R]}))\}$. Set $PL(\mathcal{B}_{[R]})$ contains pairs of the form (w', q) . First component w' is prefix of a word $w \in \mathcal{L}(\mathcal{B}_{[R]})$, but the second component q is a state of automaton $\mathcal{B}_{[R]}$ which can be reached if input word for the automaton has prefix w' .

Definition 8. $T = (\mathbf{S}, \mathcal{L}, \bar{\mathcal{L}}, \mathcal{E}, \mathcal{J})$ is a tableau for \mathcal{A} with respect to \mathcal{R} iff a) \mathbf{S} is non-empty set, b) $\mathcal{L} : \mathbf{S} \rightarrow 2^{clos(\mathcal{A})}$, c) $\bar{\mathcal{L}} : \mathbf{S} \rightarrow 2^{NFAclos(\mathcal{A}, \mathcal{R})}$, d) $\mathcal{J} : \mathcal{I}_{\mathcal{A}} \rightarrow \mathbf{S}$, e) $\mathcal{E} : \mathcal{R}_{\mathcal{A}} \rightarrow 2^{\mathbf{S} \times \mathbf{S}}$.

Furthermore, for all $C, C_1, C_2 \in clos(\mathcal{A})$; $s, t \in \mathbf{S}$; $R, S \in \mathcal{R}_{\mathcal{A}}$, and $a, b \in \mathcal{I}_{\mathcal{A}}$, the tableau T satisfies:

- (P1a) If $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$ (C is atomic, or $\exists R.Self$),
- (P1b) $\top \in \mathcal{L}(s)$, and $\perp \notin \mathcal{L}(s)$, for all s ,
- (P1c) If $\exists R.Self \in \mathcal{L}(s)$, then $\langle s, s \rangle \in \mathcal{E}(R)$,
- (P2) if $(C_1 \sqcap C_2) \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,
- (P3) if $(C_1 \sqcup C_2) \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,
- (P5) if $\exists S.C \in \mathcal{L}(s)$, then there is some t with $\langle s, t \rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(t)$,
- (P7) $\langle x, y \rangle \in \mathcal{E}(R)$ iff $\langle y, x \rangle \in \mathcal{E}(Inv(R))$,
- (P8) if $(\leq nS.C) \in \mathcal{L}(s)$, then $\sharp S^T(s, C) \leq n$,
- (P9) if $(\geq nS.C) \in \mathcal{L}(s)$, then $\sharp S^T(s, C) \geq n$,
- (P10) if $(\leq nS.C) \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(S)$, then $C \in \mathcal{L}(t)$ or $\dot{\cdot}C \in \mathcal{L}(t)$,
- (P11) if $a : C \in \mathcal{A}$, then $C \in \mathcal{L}(\mathcal{J}(a))$
- (P12) if $(a, b) : R \in \mathcal{A}$, then $(\mathcal{J}(a), \mathcal{J}(b)) \in \mathcal{E}(R)$,
- (P13) if $(a, b) : \neg R \in \mathcal{A}$, then $(\mathcal{J}(a), \mathcal{J}(b)) \notin \mathcal{E}(R)$,
- (P14) if $a \neq b \in \mathcal{A}$, then $\mathcal{J}(a) \neq \mathcal{J}(b)$,
- (P15) if $Dis(R, S) \in \mathcal{R}$, then $\mathcal{E}(R) \cap \mathcal{E}(S) = \emptyset$,
- (P16) if $\langle s, t \rangle \in \mathcal{E}(R)$ and $R \sqsubseteq S$, then $\langle s, t \rangle \in \mathcal{E}(S)$,⁵

⁵ \sqsubseteq is the transitive closure of \sqsubseteq [5]

- (P6') $\forall \mathcal{B}_{[R]}. Z \in \bar{\mathcal{L}}(s)$, where ⁶ $Z = \bigvee_{Q \in \text{comp}([R])} (Q, Z_Q, \hat{Z}_Q)$, $Z_Q = \mathcal{L}(s)|_Q = \{C | \forall Q. C \in \mathcal{L}(s)\}$ and $\hat{Z}_Q = \mathcal{L}(s) \cap \text{clos}(\mathcal{A})|_{Q^-}$, for all $s \in S$ and $[R] \in \mathcal{R}'_{\mathcal{A}}$,
- (P4a') if $\forall \mathcal{B}^p. Z \in \bar{\mathcal{L}}(s)$, $\langle s, t \rangle \in \mathcal{E}(S)$, and $p \xrightarrow{S} q \in \mathcal{B}^p$, then $\forall \mathcal{B}^q. Z \in \bar{\mathcal{L}}(t)$,
- (P4b') if $\forall \mathcal{B}^p. Z \in \bar{\mathcal{L}}(s)$, $\varepsilon \in \mathcal{L}(\mathcal{B}^p)$, and $Z = \bigvee_{j=1}^l (Q_j, Z_j, \hat{Z}_j)$ then there is j_0 , such that $Z_{j_0} \subseteq \mathcal{L}(s)$, $\mathcal{L}(s)|_{Q_{j_0}^-} \subseteq \hat{Z}_{j_0}$

where in (P8) and (P9),

$S^T(s, C) = \{t \in \mathbf{S} | \langle s, t \rangle \in \mathcal{E}(S')\}$, for some $S' \in \mathcal{L}(\mathcal{B}_S)$ and $C \in \mathcal{L}(t)$ \square .

Lemma 1. $S\mathcal{R}^{\sqcup} \mathcal{I}Q$ -Abox \mathcal{A} is consistent w.r.t. \mathcal{R} iff there exists a tableau for \mathcal{A} w.r.t. \mathcal{R} .

Proof. (\Leftarrow) Let $T = (\mathbf{S}, \mathcal{L}, \bar{\mathcal{L}}, \mathcal{E}, \mathcal{J})$ be a tableau for \mathcal{A} with respect to \mathcal{R} . An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of \mathcal{A} and \mathcal{R} can be defined as follows: $\Delta^{\mathcal{I}} := S$, $C^{\mathcal{I}} := \{s | C \in \mathcal{L}(s)\}$, for a concept name $C \in \text{clos}(\mathcal{A})$, $a^{\mathcal{I}} := \mathcal{J}(a)$ for an individual name $a \in \mathcal{I}_{\mathcal{A}}$ and for a role name $[Q] \in \mathcal{R}'_{\mathcal{A}}$, $R \in \mathcal{R}_{\mathcal{A}}$, we set $\bar{\mathcal{E}}([Q]) := \{\langle s_0, s_n \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \text{there are } s_1, \dots, s_{n-1} \text{ with } \langle s_i, s_{i+1} \rangle \in \mathcal{E}(S_{i+1}), \text{ for } 0 \leq i \leq n-1 \text{ and } S_1 S_2 \dots S_n \in \mathcal{L}(\mathcal{B}_{[Q]})\}$, $R^{\mathcal{I}} := \{\langle x, y \rangle \in \bigcup_{R \in \text{comp}([Q])} \bar{\mathcal{E}}([Q]) | \mathcal{L}(x)|_R \subseteq \mathcal{L}(y) \text{ and } \mathcal{L}(y)|_{R^-} \subseteq \mathcal{L}(x)\}$.

We have to show that \mathcal{I} is a model for \mathcal{A} and \mathcal{R} .

Next, we show that \mathcal{I} is model for \mathcal{R} . $\mathcal{I} \models \mathcal{R}_a$ can be proved by using the same method as in [5]. Let's consider a RIA of the form $R_1 \dots R_n \stackrel{\dot{}}{\sqsubseteq} T_1 \sqcup \dots \sqcup T_m$. Let's $\langle x_0, x_n \rangle \in (R_1 \dots R_n)^{\mathcal{I}}$. According to semantic rules, there are x_1, \dots, x_{n-1} such that $\langle x_i, x_{i+1} \rangle \in R_{i+1}^{\mathcal{I}}$, for $i = 0, 1, \dots, n-1$. As roles T_{ij} do not appear on the left hand side of RIAs then $R_i \in \text{comp}([Q])$ only for $Q = R_i$ i.e. $R_i^{\mathcal{I}} \subseteq \bar{\mathcal{E}}([R_i])$. This means that there are $y_{i0} = x_i, y_{i1}, \dots, y_{il_i} = x_{i+1}$ such that $\langle y_{ij}, y_{i,j+1} \rangle \in \mathcal{E}(S_{ij+1})$ and $S_{i1} \dots S_{il_i} \in \mathcal{L}(\mathcal{B}_{[R_{i+1}]})$. According to automata construction, we have the following: $S_{11} \dots S_{1l_1} S_{21} \dots S_{nl_n} \in \mathcal{L}(\mathcal{B}_{[T_1 \sqcup \dots \sqcup T_m]})$ so $\langle x_0, x_n \rangle \in \bar{\mathcal{E}}([T_1 \sqcup \dots \sqcup T_m])$. On the other side, according to rule (P6'), the following $\forall \mathcal{B}_{[T_1 \sqcup \dots \sqcup T_m]}. Z \in \bar{\mathcal{L}}(x_0)$ holds, where $Z = \bigvee_{j=1}^m (T_j, Z_{T_j}, \hat{Z}_{T_j})$. By $S_{11} \dots S_{nl_n} \in \mathcal{L}(\mathcal{B}_{[T_1 \sqcup \dots \sqcup T_m]})$ and rule (P4a') we have $\forall \mathcal{B}_{[T_1 \sqcup \dots \sqcup T_m]}. Z \in \bar{\mathcal{L}}(x_n)$ and $\varepsilon \in \mathcal{L}(\mathcal{B}_{[T_1 \sqcup \dots \sqcup T_m]}^q)$. From (P4b') we have that there is j such that $\mathcal{L}(x_0)|_{T_j} = Z_{T_j} \subseteq \mathcal{L}(x_n)$ and $\mathcal{L}(x_n)|_{T_j^-} \subseteq \hat{Z}_{T_j} \subseteq \mathcal{L}(x_0)$, i.e. $\langle x_0, x_n \rangle \in T_j^{\mathcal{I}}$. Therefore $\langle x_0, x_n \rangle \in (T_1 \sqcup \dots \sqcup T_m)^{\mathcal{I}}$.

Secondly, we prove that \mathcal{I} is model for \mathcal{A} . We show that $C \in \mathcal{L}(s)$ implies $s \in C^{\mathcal{I}}$ for each $s \in \mathbf{S}$ and each $C \in \text{clos}(\mathcal{A})$. Together with (P11)-(P14), this implies that \mathcal{I} is a model for \mathcal{A} [5]. Consider the case $C \equiv \forall R.D$. For the other cases, see [5].

Let $\forall R.D \in \mathcal{L}(s)$ and $\langle s, t \rangle \in R^{\mathcal{I}}$. If R is role name then according to definition $R^{\mathcal{I}}$ there exists $[Q]$ such that $R \in \text{comp}([Q])$, $\langle s, t \rangle \in \bar{\mathcal{E}}([Q])$ and $\mathcal{L}(s)|_R \subseteq \mathcal{L}(t)$. If $R = S^-$, where S role name, then according to definition $S^{\mathcal{I}}$ there exists role $[Q]$ such that $S \in \text{comp}([Q])$, $\langle t, s \rangle \in \bar{\mathcal{E}}([Q])$ and $\mathcal{L}(s)|_{S^-} \subseteq \mathcal{L}(t)$ (i.e. $\mathcal{L}(s)|_R \subseteq \mathcal{L}(t)$). In both cases we have $D \in \mathcal{L}(t)$. By induction, $t \in D^{\mathcal{I}}$ and thus $s \in (\forall R.D)^{\mathcal{I}}$.

⁶ Rules (P6), (P4a) and (P4b) in [5] are changed with rules (P6'), (P4a') and (P4b').

(\Rightarrow) For the converse, suppose $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model for \mathcal{A} w.r.t. \mathcal{R} . We define tableau $T = (\mathbf{S}, \mathcal{L}, \bar{\mathcal{L}}, \mathcal{E}, \mathcal{J})$ as follows:
 $\mathbf{S} := \Delta^{\mathcal{I}}$, $\mathcal{J}(a) := a^{\mathcal{I}}$, $\mathcal{E}(R) := R^{\mathcal{I}}$, $\mathcal{L}(s) := \{C \in \text{clos}(\mathcal{A}) \mid s \in C^{\mathcal{I}}\}$
 $\bar{\mathcal{L}}(s) := \{\forall \mathcal{B}_{[R]}^q.Z \mid (\exists t \in \Delta^{\mathcal{I}})(\exists w') \forall \mathcal{B}_{[R]}.Z \in \bar{\mathcal{L}}_1(t), \langle w', q \rangle \in PL(\mathcal{B}_{[R]}) \text{ and } \langle t, s \rangle \in (w')^{\mathcal{I}}\}$, where $\bar{\mathcal{L}}_1(s) := \{\forall \mathcal{B}_{[R]}.Z \mid Z = \bigvee_{Q \in \text{comp}([R])} (Q, \mathcal{L}(s)|_Q, \mathcal{L}(s) \cap \text{clos}(\mathcal{A})|_{Q^-})\}$.

We have to prove that T is tableau for \mathcal{A} w.r.t. \mathcal{R} . We restrict our attention to the only new cases. For the other cases, see [5].

The rule (P6') follows immediately from the definition of $\bar{\mathcal{L}}_1(s)$ and $\bar{\mathcal{L}}_1(s) \subseteq \bar{\mathcal{L}}(s)$ (for $t = s$ and $w' = \varepsilon$).

For (P4a'), let's $\forall \mathcal{B}_{[R]}^p.Z \in \bar{\mathcal{L}}(s)$, $\langle s, t \rangle \in \mathcal{E}(S)$. Assume that there is a transition $p \xrightarrow{S} q \in \mathcal{B}_{[R]}^p$. From definition $\bar{\mathcal{L}}(s)$ there exists $v \in \Delta^{\mathcal{I}}$ and w' such that $\forall \mathcal{B}_{[R]}.Z \in \bar{\mathcal{L}}_1(v)$, $\langle w', p \rangle \in PL(\mathcal{B}_{[R]})$ and $\langle v, s \rangle \in (w')^{\mathcal{I}}$. Let's $w'' = w'S$ then $\langle w'', q \rangle \in PL(\mathcal{B}_{[R]})$ and $\langle v, t \rangle \in (w'')^{\mathcal{I}}$, so $\forall \mathcal{B}_{[R]}^q.Z \in \bar{\mathcal{L}}(t)$.

For (P4b'), let's $\forall \mathcal{B}_{[R]}^p.Z \in \bar{\mathcal{L}}(s)$, $\varepsilon \in \mathcal{L}(\mathcal{B}_{[R]}^p)$, and $Z = \bigvee_{j=1}^l (Q_j, Z_j, \hat{Z}_j)$. By definition $\bar{\mathcal{L}}(s)$ there exists $x \in \Delta^{\mathcal{I}}$ and w' such that $\forall \mathcal{B}_{[R]}.Z \in \bar{\mathcal{L}}_1(x)$, $\langle w', q \rangle \in PL(\mathcal{B}_{[R]})$ and $\langle x, s \rangle \in (w')^{\mathcal{I}}$. Further, we have $[R] = [Q_1 \sqcup \dots \sqcup Q_l]$, $Z_j = \mathcal{L}(x)|_{Q_j}$ and $\hat{Z}_j = \mathcal{L}(x) \cap \text{clos}(\mathcal{A})|_{Q_j^-}$. By $\varepsilon \in \mathcal{B}_{[R]}^p$ we have $w' \in \mathcal{L}(\mathcal{B}_{[R]})$, so $w'^{\mathcal{I}} \subseteq (Q_1 \sqcup \dots \sqcup Q_l)^{\mathcal{I}}$, i.e. $\langle x, s \rangle \in (Q_1 \sqcup \dots \sqcup Q_l)^{\mathcal{I}}$. This means that there is j such that $\langle x, s \rangle \in Q_j^{\mathcal{I}}$. By the rules of semantics and the definition of $\mathcal{L}(s)$, we have $Z_j = \mathcal{L}(x)|_{Q_j} \subseteq \mathcal{L}(s)$ and $\mathcal{L}(s)|_{Q_j^-} \subseteq \mathcal{L}(x) \cap \text{clos}(\mathcal{A})|_{Q_j^-} = \hat{Z}_j$ \square .

Tableau algorithm for $\mathcal{SR}^{\sqcup} \mathcal{IQ}$ DL works on the completion forest on similar manner as described in [5].

Definition 9. (*Completion forest*) Completion forest for a $\mathcal{SR}^{\sqcup} \mathcal{IQ}$ -Abox \mathcal{A} and a Rbox \mathcal{R} is a labeled collection of trees $G = (V, E, \mathcal{L}, \bar{\mathcal{L}}, \neq)$ whose distinguished root nodes can be connected arbitrarily, where each node $x \in V$ is labeled with two sets $\mathcal{L}(x) \subseteq \text{clos}(\mathcal{A})$ and $\bar{\mathcal{L}}(x) \subseteq \text{NFAclos}(\mathcal{A}, \mathcal{R})$. Each edge $\langle x, y \rangle \in E$ is labeled with a set $\mathcal{L}(\langle x, y \rangle) \subseteq \mathcal{R}_{\mathcal{A}}$. Additionally, we care of inequalities between nodes in V , of the forest G , with a symmetric binary relation \neq .

If $\langle x, y \rangle \in E$, then y is called successor of the x , but x is called predecessor of y . Ancestor is the transitive closure of predecessor, and descendant is the transitive closure of successor. A node y is called an R -successor of a node x if, for some R' with $R' \sqsubseteq R$, $R' \in \mathcal{L}(\langle x, y \rangle)$. A node y is called a neighbor (R -neighbor) of a node x if y is a successor (R -successor) of x or if x is a successor ($\text{Inv}(R)$ -successor) of y . For $S \in \mathcal{R}_{\mathcal{A}}$, $x \in V$, $C \in \text{clos}(\mathcal{A})$ we define set $S^G(x, C) = \{y \mid y \text{ is } S\text{-neighbour of } x \text{ and } C \in \mathcal{L}(y)\}$

Definition 10. A completion forest G is said to contain a clash if there is a node x such that:

- $\perp \in \mathcal{L}(x)$, or
- for a concept name A , $\{A, \neg A\} \subseteq \mathcal{L}(x)$, or
- x is an S -neighbor of x and $\neg \exists S.\text{Self} \in \mathcal{L}(x)$, or

- x and y are root nodes, y is an R -neighbor of x , and $\neg R \in \mathcal{L}(\langle x, y \rangle)$, or
- there is some $Dis(R, S) \in \mathcal{R}_a$ and y is an R and an S -neighbor of x , or
- there exists a concept $(\leq nS.C) \in \mathcal{L}(x)$ and $\{y_0, \dots, y_n\} \subseteq S^G(x, C)$ with $y_i \neq y_j$ for all $0 \leq i < j \leq n$,
- there is $\forall B^p.Z \in \overline{\mathcal{L}}(x)$, with $\varepsilon \in \mathcal{L}(B^p)$, $Z = \bigvee_{j=1}^l (Q_j, Z_j, \hat{Z}_j)$ and there are no j such that $\mathcal{L}(x)|_{Q_j^-} \subseteq \hat{Z}_j$.

A completion forest that does not contain a clash is called *clash-free*. □

The blocking is employed in order to have termination [5].

Definition 11. A node is called *blocked* if it is either directly or indirectly blocked [5]. A node x is *directly blocked* if none of its ancestors are blocked, and it has ancestors x' , y and y' such that [5]:

- none of x' , y and y' is a root node,
- x is a successor of x' and y is a successor of y' , and
- $\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$, and
- $\overline{\mathcal{L}}(x) = \overline{\mathcal{L}}(y)$ and $\overline{\mathcal{L}}(x') = \overline{\mathcal{L}}(y')$, and
- $\mathcal{L}(\langle x', x \rangle) = \mathcal{L}(\langle y', y \rangle)$.

In this case we say that y *blocks* x . A node y is *indirectly blocked* if one of its ancestors is blocked [5].

The non-deterministic tableau algorithm can be described as follows:

- Input: Non-empty $SR^{\sqcup}IQ$ -Abox \mathcal{A} and a reduced Rbox \mathcal{R}
- Output: "Yes" if $SR^{\sqcup}IQ$ -Abox \mathcal{A} is consistent w.r.t. Rbox \mathcal{R} , otherwise "No"
- Method:
 1. step: Construct completion forest $G = (V, E, \mathcal{L}, \overline{\mathcal{L}}, \neq)$ as follows:
 - for each individual a occurring in \mathcal{A} , V contains a root node x_a ,
 - if $(a, b) : R \in \mathcal{A}$ or $(a, b) : \neg R \in \mathcal{A}$, then E contains an edge $\langle x_a, x_b \rangle$,
 - if $a \neq b \in \mathcal{A}$, then $x_a \neq x_b$ is in G ,
 - $\mathcal{L}(x_a) := \{C | a : C \in \mathcal{A}\}$,
 - $\overline{\mathcal{L}}(x_a) := \emptyset$,
 - $\mathcal{L}(\langle x_a, x_b \rangle) := \{R | (a, b) : R \in \mathcal{A}\} \cup \{\neg R | (a, b) : \neg R \in \mathcal{A}\}$
 Go to step 2.
 2. step: Apply an expansion rule (see table 1) to the forest G , while it is possible. Otherwise, go to step 3.
 3. step: If the forest G does not contain *clash* return "Yes", otherwise return "No".

Lemma 2. Let \mathcal{A} be a $SR^{\sqcup}IQ$ -Abox and \mathcal{R} a reduced Rbox. The tableau algorithm terminates when started for \mathcal{A} and \mathcal{R} .

Lemma 3. Let \mathcal{A} be a $SR^{\sqcup}IQ$ -Abox and \mathcal{R} a reduced Rbox. Tableau algorithm returns answer "Yes" if and only if there is a tableau for \mathcal{A} w.r.t. \mathcal{R} .

Table 1. Expansion rules for $\mathcal{SR}^{\perp}IQ$ tableau algorithm (updated from [5])

	The rules \sqcap, \sqcup, \exists , Self, \leq_r, \geq, \leq are defined in [5], but only in rules that create new node y should set $\bar{\mathcal{L}}(y) := \emptyset$
ch'	If x is not indirectly blocked and there is concept $C \in \text{clos}(\mathcal{A})$ with $\{C, \dot{\neg}C\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{E\}$, for some $E \in \{C, \dot{\neg}C\}$
\forall'_1	If x is not indirectly blocked and it is not possible to apply ch' -rule to $\mathcal{L}(x)$, and $\forall \mathcal{B}_{[R]}.Z \notin \bar{\mathcal{L}}(x)$, where $Z = \bigvee_{Q \in \text{comp}([R])} (Q, \mathcal{L}(x) _Q, \mathcal{L}(x) \cap \text{clos}(\mathcal{A}) _{Q^-})$ then $\bar{\mathcal{L}}(x) \rightarrow \bar{\mathcal{L}}(x) \cup \{\forall \mathcal{B}_{[R]}.Z\}$
\forall'_2	If $\forall \mathcal{B}^p.Z \in \bar{\mathcal{L}}(x)$, and x is not indirectly blocked, $p \xrightarrow{S} q \in \mathcal{B}^p$ and there is S-neighbor y of x with $\forall \mathcal{B}^q.Z \notin \bar{\mathcal{L}}(y)$ then $\bar{\mathcal{L}}(y) \rightarrow \bar{\mathcal{L}}(y) \cup \{\forall \mathcal{B}^q.Z\}$
\forall'_3	If $\forall \mathcal{B}^p.Z \in \bar{\mathcal{L}}(y)$, and y is not indirectly blocked, $\varepsilon \in \mathcal{L}(\mathcal{B}^p)$, $Z = \bigvee_{j=1}^l (Q_j, Z_j, \hat{Z}_j)$ and there is no j such that $Z_j \subseteq \mathcal{L}(y)$ and $\mathcal{L}(y) _{Q_j^-} \subseteq \hat{Z}_j$ then choose j such that $\mathcal{L}(y) _{Q_j^-} \subseteq \hat{Z}_j$ and $\bar{\mathcal{L}}(y) \rightarrow \bar{\mathcal{L}}(y) \cup Z_j$.

Proof. For the if direction, suppose that the algorithm returns "Yes". It means that the algorithm generated completion forest $G = (V, E, \mathcal{L}, \bar{\mathcal{L}}, \neq)$ without clash and there are no expansion rules (see table 1) that can be applied.

Let's $b(x) = x$, if x is not blocked and $b(x) = y$, if y blocks node x .

A path [6] is a sequence of pairs nodes of \mathbf{G} of the form

$$p = \langle (x_0, x'_0), \dots, (x_n, x'_n) \rangle. \quad (5)$$

For such a path, we define $Tail(p) = x_n$ and $Tail'(p) = x'_n$. We denote the path

$$\langle (x_0, x'_0), (x_1, x'_1), \dots, (x_n, x'_n), (x_{n+1}, x'_{n+1}) \rangle \quad (6)$$

with $\langle p|(x_{n+1}, x'_{n+1}) \rangle$. The set of $Paths(\mathbf{G})$ can be defined inductively as follows:

- if x_0 is root node then $\langle x_0, x_0 \rangle \in Paths(\mathbf{G})$
- if $p \in Paths(\mathbf{G})$, $z \in V$ and z is not indirectly blocked, such that $\langle Tail(p), z \rangle \in E$, then $\langle p, (b(z), z) \rangle \in Paths(\mathbf{G})$

We define structure $T = (\mathbf{S}, \mathcal{L}, \bar{\mathcal{L}}, \mathcal{E}, \mathcal{J})$ as follows $\mathbf{S} := Paths(\mathbf{G})$, $\mathcal{L}(p) := \mathcal{L}(Tail(p))$, $\bar{\mathcal{L}}(p) := \bar{\mathcal{L}}(Tail(p))$, if root node x_a denotes individual a then $\mathcal{J}(a) = (\langle x_a, x_a \rangle)$ and $\mathcal{E}(R) := \{(s, t) \in \mathbf{S} \times \mathbf{S} | t = (s, \langle b(y), y \rangle) \text{ and } y \text{ is an } R\text{-successor of } Tail(s) \text{ or } s = (t, \langle b(y), y \rangle) \text{ and } y \text{ is an } Inv(R)\text{-successor of } Tail(t)\} \cup \{\langle \mathcal{J}(a), \mathcal{J}(b) \rangle | x_b \text{ is an } R\text{-neighbour of } x_a\}$.

Thus defined structure T is a tableau. New rules (P6'), (P4a') directly follows from \forall'_1 and \forall'_2 rule, but (P4b') follows from \forall'_3 and definition of clash (see definition (10)). For the other cases, see [6].

For the only-if direction, the proof is the same as proof in [4, 5] (i.e., we take a tableau and use it to steer the application of the non-deterministic rules). \square

From Theorem 1 in [5] and Lemmas 1, 2 and 3, we thus have the following theorem:

Theorem 1. *The tableau algorithm decides satisfiability and subsumption of $SR^{\perp}IQ$ -concepts with respect to Aboxes, Rboxes, and Tboxes.*

4 Conclusion

It is important to note that original idea of extension \mathcal{ALC} DL with composition-based RIAs is presented in [11]. We introduce more expressive formalism that allows composition-based RIAs and relaxed restrictions defined in [11]. Motivated by practical applications in manufacturing engineering we define tableau algorithm in order to check satisfiability of $SR^{\perp}IQ$ DL. Future research will be focused on how to extend regularity conditions for $SRQIQ$ DL in order to support composition-based RIAs as well as at the same time support RIAs proposed in [9]. We use the algorithm proposed in this paper for modeling the regulations of capital adequacy of credit institutions.

References

1. Baader, F. Calvanese, D., McGuinness, D., Nardi D., Patel-Schneider, P.: The description logic handbook - theory, implementation and application. Cambridge University Press, second edition (2007)
2. Bock, C., Zha, X., Suh, H., Lee, J.: Ontological product modeling for collaborative design. *Advanced Engineering Informatics* **24** (2010) 510-524
3. Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Schneider, P. P., Sattler, U.: OWL 2: The next step for OWL, *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* **6**(4) (2008) 309-322
4. Horrocks, I., Sattler, U.: Decidability of $SHIQ$ with complex role inclusion axioms. *Artificial Intelligence* **160**(1-2) (2004) 79-104
5. I. Horrocks, O. Kutz, and U. Sattler. The irresistible $SRIQ$. In *Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions and Technical Report, 2005*. <http://www.cs.man.ac.uk/~sattler/publications/sriq-tr.pdf>
6. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $SRQIQ$. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*. (2006) 57-67
7. Kazakov, Y.: An extension of complex role inclusion axioms in the description logic $SRQIQ$. In *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR 2010)*, Edinburgh, UK. (2010) 472-487
8. Krdžavac, N., Bock, C.: Reasoning in manufacturing part-part examples with OWL2. U.S. National Institute of Standards and Technology, Technical Report NISTIR 7535. (2008)
9. Mosurović, M., Krdžavac, N.: A technique for handling the right hand side of complex RIA. In *Proc. of 24th International Workshop on Description Logics (DL2011)*, Barcelona, Spain. (2011) 543-554
10. Schmidt-Schau, M.: Subsumption in KL-ONE is undecidable. In *Principle of Knowledge Representation and Reasoning-Proceedings of the First International Conference KR89*. (1989)
11. Wessel, M.: Obstacles on the way to spatial reasoning with description logics - some undecidability results. In *Proceedings of the International Workshop on Description Logics 2001 (DL 2001)*, CEUR Workshop Proceedings. Volume 49., (2001)

Optimising Parallel ABox Reasoning of \mathcal{EL} Ontologies*

Yuan Ren¹, Jeff Z. Pan¹ and Kevin Lee²

¹University of Aberdeen, Aberdeen, UK

²NICTA, Australia

Abstract. The success of modern multi-core processors makes it possible to develop parallel ABox reasoning algorithms to facilitate efficient reasoning on large scale ontological data sets. In this paper, we extend a parallel TBox reasoning algorithm for $\mathcal{ELH}_{\mathcal{R}+}$ to a parallel ABox reasoning algorithm for $\mathcal{ELH}_{\perp, \mathcal{R}+}$, which also supports the bottom concept so as to model disjointness and inconsistency. In design of algorithms, we exploit the characteristic of ABox reasoning in $\mathcal{ELH}_{\perp, \mathcal{R}+}$ to improve parallelisation and reduce unnecessary resource cost. Particularly, we separate the TBox reasoning, ABox reasoning on types and ABox reasoning on relations. Our evaluation shows that a naive implementation of our approach can compute all ABox entailments of a Not-Galen⁻ ontology with about 1 million individuals and 9 million axioms in about 3 minutes.

1 Introduction

Optimisation of reasoning algorithms is one of the core research topics in description logic (DL) study. In the last decades, highly-efficient DL reasoning systems have been implemented with different optimisation technologies. So far, these systems are designed for a single computation core. Reasoning is performed sequentially and can not be parallelised. With the development of modern computing hardware, it is possible and also desired to parallelise reasoning procedures to improve efficiency and scalability.

One direction of parallel reasoning is to use a cluster of multiple computer nodes (or simply, *peers*). In Marvin [9], peers use a divide-conquer-swap strategy for RDFS inference. Weaver and Handler propose a parallel RDFS inference engine [18], in which peers use an ABox partitioning approach. In SAOR [2], peers use optimised template rules for join-free inference in pD* [3]. In DRAGO [13], peers perform OWL DL reasoning under the setting of Distributed Description Logics. A distributed resolution algorithm for \mathcal{ALC} was proposed by Schlicht and Stuckenschmidt [11] and further improved and extended to \mathcal{ALCHIQ} [12]. MapReduce has also been adopted to support ABox reasoning in RDFS [17], pD* [16] as well as justifications in pD* [19], and TBox reasoning in \mathcal{EL}^+ [7] (there is no implementation for the \mathcal{EL}^+ case yet).

Another direction of parallel reasoning is to use multiple computation cores (or simply, *workers*) in a single computer. Soma and Prasanna [14] propose to use data-partitioning and rule-partitioning in their parallel algorithms for pD*. Liebig and Müller exploit the non-determinism introduced by disjunctions or number restrictions in the \mathcal{SHN} tableau algorithm [6], so that multiple workers can apply expansion rules on

* This paper is an extended version of “Parallel ABox Reasoning of \mathcal{EL} Ontologies” [10].

independent alternatives. Similarly, Meissner [8] proposes parallel expansions of independent branchings in an \mathcal{ALC} tableau and experimented with 3 different strategies. Aslani and Haarslev [1] propose a parallel algorithm for OWL DL classification. Recently, Kazakov et al. [4] present a lock-free parallel completion-based TBox classification algorithm for $\mathcal{ELH}_{\mathcal{R}_+}$. They later extend this work to support nominals [5] but the impact on parallelisation has not been reported.

In this paper, we extend the parallel TBox reasoning algorithm [4] for $\mathcal{ELH}_{\mathcal{R}_+}$ to a parallel and lock-free ABox reasoning algorithm for $\mathcal{ELH}_{\perp, \mathcal{R}_+}$, which also supports the bottom concept so as to model disjointness and inconsistency. We will optimise the parallelisation by separating TBox classification, the computation of types and relations for individuals. We show that our completion rules and algorithms are complete and sound. Our evaluation shows that a naive implementation of our approach can achieve high performance and scalability. Comparing to the original version [10], this paper extends with new optimisations (particularly, Sect. 4.2 and Sect. 4.3) and evaluation regarding ABox reasoning.

The remainder of the paper is organised as follows: In Sect. 2 we introduce background knowledge of DLs $\mathcal{ELH}_{\mathcal{R}_+}$ and $\mathcal{ELH}_{\perp, \mathcal{R}_+}$, and the parallel $\mathcal{ELH}_{\mathcal{R}_+}$ TBox classification algorithm [4]. In Sect. 3 we explain the technical challenges, before presenting the completion rules and parallel ABox reasoning algorithms for $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ in Sect. 4. We evaluate our approach in Sect. 5, before we conclude the paper in Sect. 6.

The proof of all lemmas and theorems are included in our online tech report at <http://www.box.com/s/3636a703614b65f6cdba>.

2 Preliminary

2.1 DL $\mathcal{ELH}_{\mathcal{R}_+}$ and $\mathcal{ELH}_{\perp, \mathcal{R}_+}$

A *signature* of an ontology \mathcal{O} is a triple $\Sigma_{\mathcal{O}} = (\mathcal{CN}_{\mathcal{O}}, \mathcal{RN}_{\mathcal{O}}, \mathcal{IN}_{\mathcal{O}})$ consisting of three mutually disjoint finite sets of atomic concepts $\mathcal{CN}_{\mathcal{O}}$, atomic roles $\mathcal{RN}_{\mathcal{O}}$ and individuals $\mathcal{IN}_{\mathcal{O}}$. Given a signature, complex concepts in $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ can be defined inductively using the $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ constructors as in Table 1. $\mathcal{ELH}_{\mathcal{R}_+}$ supports all $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ constructors except \perp . Two concepts C and D are equivalent if they mutually include each other, denoted by $C \equiv D$. An ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ consists of a TBox \mathcal{T} and an ABox \mathcal{A} , which are finite sets of TBox axioms and ABox axioms, respectively. $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ allows all axioms listed in Table 1. $\mathcal{ELH}_{\mathcal{R}_+}$ allows all except individual inequalities. The semantics of constructors and axioms are also listed in Table 1. Given an ontology \mathcal{O} , we use $\sqsubseteq_{\mathcal{O}}^*$ to represent the reflexive transitive closure of RIs. It is easy to see that in an $\mathcal{ELH}_{\mathcal{R}_+} / \mathcal{ELH}_{\perp, \mathcal{R}_+}$ ontology \mathcal{O} , all of such $\sqsubseteq_{\mathcal{O}}^*$ relations can be computed in polynomial time w.r.t. the size of \mathcal{O} . In ABox reasoning, we are particularly interested in ABox materialisation, i.e. finding all $A(a)$ s.t. $a \in \mathcal{IN}_{\mathcal{O}}$, $A \in \mathcal{CN}_{\mathcal{O}}$, $\mathcal{O} \models A(a)$ and all $r(a, b)$ s.t. $a, b \in \mathcal{IN}_{\mathcal{O}}$, $r \in \mathcal{RN}_{\mathcal{O}}$ and $\mathcal{O} \models r(a, b)$. Such results can be very useful for efficient on-line instance retrieval and/or query answering.

2.2 Parallel TBox Classification of $\mathcal{ELH}_{\mathcal{R}_+}$ Ontologies

Given an ontology \mathcal{O} , TBox classification is a reasoning task that computes all inclusions over atomic concepts in \mathcal{O} . Kazakov et al. [4] proposed an approach to parallel

Table 1. $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ syntax and semantics

Concepts:		
atomic concept	A	$A^{\mathcal{I}}$
top	\top	$\Delta^{\mathcal{I}}$
bottom	\perp	\emptyset
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \mid \exists y. \langle x, y \rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
Roles:		
atomic role	r	$r^{\mathcal{I}}$
TBox Axioms:		
general concept inclusion (GCI):	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion (RI):	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
role transitivity:	$Trans(t)$	$t^{\mathcal{I}} \times t^{\mathcal{I}} \subseteq t^{\mathcal{I}}$
ABox Axioms:		
class assertion:	$A(a)$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
role assertion:	$r(a, b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$
individual equality:	$a \doteq b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
individual inequality:	$a \not\doteq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

TBox classification for $\mathcal{ELH}_{\mathcal{R}_+}$. They devise a set of completion rules as follows.

$$\begin{array}{l}
 \mathbf{R}_{\sqsubseteq} \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{O} \\
 \mathbf{R}_{\top}^+ \frac{C \sqsubseteq C}{C \sqsubseteq \top} : \top \text{ occurs in } \mathcal{O} \\
 \mathbf{R}_{\sqcap}^+ \frac{C \sqsubseteq D_1, C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2} : D_1 \sqcap D_2 \text{ occurs in } \mathcal{O} \\
 \mathbf{R}_{\exists}^+ \frac{C \sqsubseteq D}{\exists s.C \rightarrow \exists s.D} : \exists s.D \text{ occurs in } \mathcal{O} \\
 \mathbf{R}_{\mathcal{H}} \frac{D \sqsubseteq \exists r.C, \exists s.C \rightarrow E}{D \sqsubseteq E} : r \sqsubseteq_{\mathcal{O}}^* s \\
 \mathbf{R}_T \frac{D \sqsubseteq \exists r.C, \exists s.C \rightarrow E}{\exists t.D \rightarrow E} : r \sqsubseteq_{\mathcal{O}}^* t \sqsubseteq_{\mathcal{O}}^* s, Trans(t) \in \mathcal{O} \\
 \mathbf{R}_{\sqcap}^- \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1; C \sqsubseteq D_2} \\
 \mathbf{R}_{\exists}^- \frac{C \sqsubseteq \exists R.D}{D \sqsubseteq D}
 \end{array}$$

In the above rules, $D \rightarrow E$ denotes the special form of GCIs where D and E are both existential restrictions. Given an $\mathcal{ELH}_{\mathcal{R}_+}$ ontology \mathcal{O} that has no ABox, these rules infer $C \sqsubseteq D$ iff $\mathcal{O} \models C \sqsubseteq D$ for all C and D such that $C \sqsubseteq C \in \mathbf{S}$ and D occurs in \mathcal{O} [4, Theorem 1], where \mathbf{S} is the set of axioms closed under the above rules. The completion rules are designed in a way that *all premises of each rule have a common concept* (the concept C in each rule), which is called a *context* of the corresponding premise axiom(s). Each context maintains a queue of axioms called *scheduled*, on which some completion rule can be applied, and a set of axioms called *processed*, on which some completion rule has already been applied. *An axiom can only be included in the scheduled queues and/or processed sets of its own contexts.* To ensure that multiple

workers can share the queues and sets without locking them, they further devised a concurrency mechanism, in which: (i) each worker will process a single context at a time and vice versa; (ii) the processing of all axioms in the scheduled queue of a context requires no axioms from the processed sets of other contexts. To realise all these, all contexts with non-empty schedules are arranged in a queue called *activeContexts*. A context can be added into the *activeContexts* queue only if it is not already in the queue.

Here are the key steps of the parallel TBox algorithm:

1. Tautology axiom $A \sqsubseteq A$ for each $A \in \mathcal{CN}_{\mathcal{O}}$ is added to the scheduled queue of A . All active contexts are added into the queue of *activeContexts*.
2. Every idle worker always looks for the next context in the *activeContexts* queue and processes axioms in its scheduled queue.
 - (a) Pop an axiom from the scheduled queue, add it into the processed set of the context.
 - (b) Apply completion rules to derive conclusions.
 - (c) Add each derived conclusion into the scheduled queue of its corresponding context(s), which will be activated if possible.

Before we extend the parallel TBox reasoning algorithm to support ABox reasoning in Sect. 4, we first discuss the challenges to deal with in parallel ABox reasoning.

3 Technical Challenges: Parallel ABox Reasoning

A naive approach to ABox materialisation is to internalise the entire ABox into TBox (i.e., converting assertions of form $C(a)$ into $\{a\} \sqsubseteq C$, and $R(a, b)$ into $\{a\} \sqsubseteq \exists R.\{b\}$) and treat the internalised “nominals” as atomic concepts with TBox classification rules. This is inefficient due to unnecessary computation and maintenance costs. For example, axiom $\{a\} \sqsubseteq \exists r.C$ has C as a context. Thus once derived, it will be maintained in the *processed* set of C , as a possible left premise of Rule $\mathbf{R}_{\mathcal{H}}$ and/or \mathbf{R}_T . However, $\mathcal{ELH}_{\perp, \mathcal{R}+}$ does not support nominals, meaning that any corresponding right premise $\exists s.C \rightarrow E$ can always be computed independently from (or before) the derivation of $\{a\} \sqsubseteq \exists r.C$. Therefore it is unnecessary to maintain $\{a\} \sqsubseteq \exists r.C$ in context C because with all possible right premises pre-computed, it can be directly used to trigger all rules.

Even with TBox and ABox reasoning separated, performance and scalability can still be improved: (1) The computation of relations in $\mathcal{ELH}_{\perp, \mathcal{R}+}$ is independent from the computation of types (Lemma 2). Thus when crafting type reasoning rules, relations can be used as side conditions instead of premises. (2) Among all the relations that can be entailed by an $\mathcal{ELH}_{\perp, \mathcal{R}+}$ ontology, some can be trivially computed and do not contribute to type computation. These relations can be easily recovered in retrieval (Lemma 3). (3) Without individual equality, the computation of relations in $\mathcal{ELH}_{\perp, \mathcal{R}+}$ can be perfectly parallelised on an individual basis. This indicates that when computing relations, the concurrency mechanism can be simplified. As we will show, we no longer need to maintain the *scheduled* queue and this improves scalability and performance.

The above optimisations are related to the design of completion rules. For further optimisation on the execution of algorithms, we refer readers to our original paper [10].

4 Approach

4.1 TBox Completion Rules

We first extend the $\mathcal{ELH}_{\mathcal{R}+}$ TBox completion rules to support the bottom concept:

$$\mathbf{R}_{\perp} \frac{D \sqsubseteq \exists r.C, C \sqsubseteq \perp}{D \sqsubseteq \perp}$$

In what follows, we call the set containing the above rule and the $\mathcal{ELH}_{\mathcal{R}+}$ rules in Sect. 2.2 the \mathbf{R} rule set, which is sound and complete for $\mathcal{ELH}_{\perp, \mathcal{R}+}$ classification:

Lemma 1. *For an $\mathcal{ELH}_{\mathcal{R}+}$ TBox \mathcal{O} , let S be any set of TBox axioms closed under the \mathbf{R} rule set, using \mathcal{O} axioms as side conditions, and $\perp \sqsubseteq \perp \in S$ if \perp occurs in \mathcal{O} . Then for any C and D such that $C \sqsubseteq C \in S$, D occurs in \mathcal{O} , we have $\mathcal{O} \models C \sqsubseteq D$ iff $C \sqsubseteq D \in S$ or $C \sqsubseteq \perp \in S$.*

The \leftarrow direction is trivial. The \rightarrow direction can be proved with contrapositive. Assuming there are $X \sqsubseteq X \in S$ and Y occurs in \mathcal{O} s.t. $X \sqsubseteq Y \notin S$ and $X \sqsubseteq \perp \notin S$, we construct a model of \mathcal{O} based on S and shows that this model does not satisfy $X \sqsubseteq Y$.

With the \mathbf{R} rules we can perform TBox reasoning:

Definition 1. (TBox Completion Closure) *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\perp, \mathcal{R}+}$ ontology, its TBox completion closure, denoted by $S_{\mathcal{T}}$, is the smallest set of axioms closed under rule set \mathbf{R} , using \mathcal{O} axioms as side conditions, such that: for all $A \in \mathcal{CN}_{\mathcal{O}}$, $A \sqsubseteq A \in S_{\mathcal{T}}$; $\perp \sqsubseteq \perp \in S_{\mathcal{T}}$ if \perp occurs in \mathcal{O} .*

According to Lemma 1, we have $A \sqsubseteq C \in S_{\mathcal{T}}$ or $A \sqsubseteq \perp \in S_{\mathcal{T}}$ for any A and C where A is an atomic concept and C occurs in \mathcal{T} . This realises TBox classification.

4.2 Relation Completion Rules

As we pointed out in Sect. 3, relations can be computed independently from types. This is characterised by the following lemma.

Lemma 2. *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\perp, \mathcal{R}+}$ ontology and $\mathcal{A}_t \subseteq \mathcal{A}$ be the set of all class assertions. Then \mathcal{O} is inconsistent, or for all $r \in \mathcal{RN}_{\mathcal{O}}$, $a, b \in \mathcal{IN}_R$, we have $\mathcal{O} \models r(a, b)$ iff $\mathcal{O} \setminus \mathcal{A}_t \models r(a, b)$.*

Now we present the ABox completion rules for $\mathcal{ELH}_{\perp, \mathcal{R}+}$. Although $\mathcal{ELH}_{\perp, \mathcal{R}+}$ does not support nominals ($\{a\}$), we still denote individuals with nominals since this helps simplify the presentation: (i) ABox rules are more readable, as they have similar syntactic forms to the TBox ones, and (ii) some of the ABox rules can be unified. More precisely, we establish the following mappings as syntactic sugar:

$$\begin{aligned} C(a) &\Leftrightarrow \{a\} \sqsubseteq C, & a \doteq b &\Leftrightarrow \{a\} \equiv \{b\}, \\ a \neq b &\Leftrightarrow \{a\} \sqcap \{b\} \sqsubseteq \perp, & r(a, b) &\Leftrightarrow \{a\} \sqsubseteq \exists r.\{b\}, \end{aligned}$$

Obviously, these mappings are semantically equivalent. In the rest of the paper, without further explanation, we treat the LHS and RHS of each of the above mappings as a *syntactic* variation of each other.

We first present the relation completion rules as follows:

$$\begin{aligned} \mathbf{AR}_{\mathcal{H}}^R \frac{\{b\} \sqsubseteq \exists r.\{a\}}{\{d\} \sqsubseteq \exists r.\{c\}} : \{a\} \sqsubseteq \{c\}, \{d\} \sqsubseteq \{b\} \in \mathcal{A} \\ \mathbf{AR}_{\mathcal{T}}^R \frac{\{b\} \sqsubseteq \exists r.\{a\}}{\{c\} \sqsubseteq \exists s.\{a\}} : \{c\} \sqsubseteq \exists t.\{b\} \in \mathcal{A}, r, t \sqsubseteq_{\mathcal{O}}^* s, \text{Trans}(s) \in \mathcal{O} \end{aligned}$$

It is worth noting that (1) without individual equality, the $\mathbf{AR}_{\mathcal{H}}^R$ rule is not needed, rendering the $\mathbf{AR}_{\mathcal{T}}^R$ rule perfectly parallelisable. This is an important property because in $\mathcal{ELH}_{\perp, \mathcal{R}+}$ ontologies, individual equality can be easily eliminated by pre-computing equal individuals and using one of them as a representative; (2) relation computation is also independent from TBox completion. We call the reasoning results with the above rules the relation completion closure:

Definition 2. (Relation Completion Closure) Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\perp, \mathcal{R}+}$ ontology, its relation completion closure, denoted by $S_{\mathcal{R}}$, is the smallest set of axioms closed under the rules $\mathbf{AR}_{\mathcal{H}}^R$ and $\mathbf{AR}_{\mathcal{T}}^R$, using \mathcal{O} axioms as side conditions, such that $S_{\mathcal{T}} \subseteq S_{\mathcal{R}}$ and $\mathcal{A} \setminus \mathcal{A}_t \subseteq S_{\mathcal{R}}$ (axioms mapped as elaborated before), where $\mathcal{A}_t \subseteq \mathcal{A}$ is the set of all class assertions.

The soundness and tractability of relation completion closure is quite obvious. Its completeness can be characterised by the following lemma.

Lemma 3. For any $\mathcal{ELH}_{\perp, \mathcal{R}+}$ ontology \mathcal{O} , let $S_{\mathcal{R}}$ be its relation completion closure, then \mathcal{O} is inconsistent, or $\mathcal{O} \models r(a, b)$ only if $\{a\} \sqsubseteq \exists s.\{b\}, s \sqsubseteq_{\mathcal{O}}^* r \in S_{\mathcal{R}}$ for $r \in \mathcal{RN}_{\mathcal{O}}$.

4.3 Type Completion Rules

Now we present the other ABox completion rules as follows, which should be applied *after* a complete closure $S_{\mathcal{R}}$ is constructed. In contrast to the **R** rules, the following rules contain concepts $D_{(i)}$ and E that can take multiple forms including nominals. Thus the mapping between ABox and TBox axioms allows us to describe the rules in a more compact manner. Together with the above two relation completion rules, we call all the ABox completion rules the **AR** rules.

The **AR** rules deserve some explanations: There are clear correspondences between the **R** rules and **AR** rules. For example, $\mathbf{AR}_{\sqsubseteq}$ is an ABox counterpart of \mathbf{R}_{\sqsubseteq} except that the context is explicitly a nominal, and TBox results are used as side conditions. Note that directly applying the **R** rules together with the **AR** rules could introduce unnecessary performance overheads such as axiom scheduling, processing and maintenance as we discussed in Sect. 3. In our approach, we separate TBox reasoning, ABox relation computation and ABox type computation. This helps reduce memory usage and computation time.

$$\begin{aligned}
\mathbf{AR}_{\sqsubseteq} & \frac{\{a\} \sqsubseteq D}{\{a\} \sqsubseteq E} : D \sqsubseteq E \in S_{\mathcal{R}} \cup \mathcal{A} \\
\mathbf{AR}_{\mathcal{H}}^* & \frac{\{a\} \sqsubseteq \exists r.D}{\{a\} \sqsubseteq E} : \exists s.D \rightarrow E \in S_{\mathcal{R}}, r \sqsubseteq_{\mathcal{O}}^* s \\
\mathbf{AR}_{\mathcal{T}}^* & \frac{\{a\} \sqsubseteq \exists r.D}{\exists t.\{a\} \rightarrow E} : \exists s.D \rightarrow E \in S_{\mathcal{R}}, r \sqsubseteq_{\mathcal{O}}^* t \sqsubseteq_{\mathcal{O}}^* s, \text{Trans}(t) \in \mathcal{O} \\
\mathbf{AR}_{\sqcap}^- & \frac{\{a\} \sqsubseteq D_1 \sqcap D_2}{\{a\} \sqsubseteq D_1; \{a\} \sqsubseteq D_2} \\
\mathbf{AR}_{\sqcap}^+ & \frac{\{a\} \sqsubseteq D_1, \{a\} \sqsubseteq D_2}{\{a\} \sqsubseteq D_1 \sqcap D_2} : D_1 \sqcap D_2 \text{ occurs in } \mathcal{O} \\
\mathbf{AR}_{\exists}^+ & \frac{\{a\} \sqsubseteq D}{\exists s.\{a\} \rightarrow \exists s.D} : r \sqsubseteq_{\mathcal{O}}^* s, \exists r.D \text{ occurs in } \mathcal{O} \\
\mathbf{AR}_{\perp} & \frac{\{a\} \sqsubseteq \perp}{\{b\} \sqsubseteq \perp} : \{b\} \sqsubseteq \exists r.\{a\} \in S_{\mathcal{R}} \\
\mathbf{AR}_{\perp}^* & \frac{\{a\} \sqsubseteq \exists r.D}{\{a\} \sqsubseteq \perp} : D \sqsubseteq \perp \in S_{\mathcal{R}} \\
\mathbf{AR}_{\mathcal{H}} & \frac{\exists s.\{a\} \rightarrow E}{\{b\} \sqsubseteq E} : r \sqsubseteq_{\mathcal{O}}^* s, \{b\} \sqsubseteq \exists r.\{a\} \in S_{\mathcal{R}} \\
\mathbf{AR}_{\mathcal{T}} & \frac{\exists s.\{a\} \rightarrow E}{\exists t.\{b\} \rightarrow E} : r \sqsubseteq_{\mathcal{O}}^* t \sqsubseteq_{\mathcal{O}}^* s, \text{Trans}(t) \in \mathcal{O}, \{b\} \sqsubseteq \exists r.\{a\} \in S_{\mathcal{R}},
\end{aligned}$$

Now we defined the closure of applying all the rules:

Definition 3. (Ontology Completion Closure) Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ ontology, its ontology completion closure, denoted by \mathbf{S} , is the smallest set of axioms closed under the \mathbf{AR} rule set, with \mathcal{O} axioms as side conditions, such that $S_{\mathcal{R}} \subseteq \mathbf{S}$; $\mathcal{A} \subseteq \mathbf{S}$ (axioms mapped as elaborated at the beginning of this section); and for all $a \in \mathcal{IN}_{\mathcal{O}}$, $\{a\} \sqsubseteq \{a\} \in X_{\mathcal{O}}$, and $\{a\} \sqsubseteq \top$ if \top occurs in \mathcal{O} .

Together the \mathbf{AR} rules are also complete, sound and tractable. The soundness and tractability of rules are quite obvious. The completeness on ABox materialisation can be shown by the following Theorem:

Theorem 1. For any $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, we have either there is some $\{x\} \sqsubseteq \perp \in \mathbf{S}$, or

1. $\mathcal{O} \models D(a)$ only if $\{a\} \sqsubseteq D \in \mathbf{S}$ for D occurs in \mathcal{O} ;
2. $\mathcal{O} \models r(a, b)$ only if $\{a\} \sqsubseteq \exists s.\{b\}, s \sqsubseteq_{\mathcal{O}}^* r \in \mathbf{S}$ for $r \in \mathcal{RN}_{\mathcal{O}}$.

The result regarding roles is quite obvious due to Lemma 3 and item 1 in Def. 3. The type part can be proved by contrapositive. Assuming there is no $\{x\} \sqsubseteq \perp \in \mathbf{S}$, it's obvious that the TBox \mathcal{T} has a model. We can construct such a model based on $S_{\mathcal{T}}$ and

shows that it can be extended to a model of \mathcal{O} based on \mathbf{S} , such that this model entails a said class assertion only if it is in \mathbf{S} . Full proof can be found in our technical report.

As we can see, the **AR** rules also preserve the feature that all premises of each rule have a same common part as the context. Therefore, they still enjoy the lock-free feature in reasoning. In later sections, we will further elaborate this point.

4.4 Parallel Algorithms

In this section, we present the parallel algorithm corresponding to the $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ completion rules. We reuse some notions such as *context*, *activeContexts* queue, *scheduled* queue and *processed* set from the original TBox algorithm for $\mathcal{ELH}_{\mathcal{R}^+}$ [4] to realise the lock-free mechanism. Axioms are indexed w.r.t. corresponding contexts.

The revised saturation algorithm (Algorithm 1) is presented as follows. The saturation of an ontology is realised by first performing saturation of the non-class assertion axioms, the output of which (i.e., $S_{\mathcal{R}}$) is then used in the saturation of the class assertions. This yields \mathbf{S} , which satisfies Theorem 1. This result contains all entailed class assertions, and all role assertions can be easily retrieved. All necessary tautology axioms must be added to the input prior to saturation. For TBox, we add axioms of the form $C \sqsubseteq C$ for all concepts C such that $C \in \mathcal{CN}_{\mathcal{O}} \cup \{\perp\}$. Similarly, for type computation we add $\{a\} \sqsubseteq \{a\}$ for all individuals $a \in \mathcal{IN}_{\mathcal{O}}$.

Algorithm 1: saturate(input): saturation of axioms under inference rules

Input: input (the set of input axioms)
Result: the saturation of input is computed in context.processed

```

1 activeContexts  $\leftarrow$   $\emptyset$ ;
2 axiomQueue.addAll(input);
3 loop
4   axiom  $\leftarrow$  axiomQueue.pop();
5   if axiom = null then break;
6   for context  $\in$  getContexts(axiom) do
7     context.scheduled.add(axiom);
8     activeContexts.activate(context);
9 loop
10  context  $\leftarrow$  activeContexts.pop();
11  if context = null then break;
12  loop
13    axiom  $\leftarrow$  context.scheduled.pop();
14    if axiom = null then break;
15    process (axiom);
16  context.isActive  $\leftarrow$  false;
17  if context.scheduled  $\neq$   $\emptyset$  then activeContexts.activate(context);

```

In the saturation (Algorithm 1), the *activeContexts* queue is initialised with an empty set (line 1), and then all input axioms are added into an *axiomQueue* (line 2). After that, two main loops (lines 3-8 and lines 9-17) are sequentially parallelised.

In the first main loop, multiple workers independently retrieve axioms from the *axiomQueue* (line 4), then get the contexts of the axioms (line 6), add the axioms into corresponding *scheduled* queues (line 7) and activate the contexts. In the second main loop, multiple workers independently retrieve contexts from the *activeContexts* queue (line 10) and process its *scheduled* axioms (line 15). Once *context.scheduled* is empty, *context.isActive* is set to *false* (line 16). A re-activation checking is performed (line 17) in case other workers have added new axioms into *context.scheduled* while the last axiom is being processed (between line 14 and line 16). This procedure will continue until the *activeContexts* queue is empty.

The *getContext()* method returns the context of an axiom, depending on the form of the axiom. In the following list, the concept *C* or $\{a\}$ is the context.

$$C \sqsubseteq D \mid D \sqsubseteq \exists r.C \mid \exists s.C \rightarrow E \mid D \sqsubseteq \exists r.\{a\} \mid \{a\} \sqsubseteq D$$

To activate a context, an atomic boolean value *isActive* is associated with each context to indicate whether the context is already active. A context is added into the *activeContexts* queue only if this value is *false*, which will be changed to *true* at the time of activation. The *process()* method covers items 2.(a), 2.(b), 2.(c) shown at the end of Sect. 2.2. We match the form of input axiom and check whether it has been processed before; if not it will be added into the *processed* set of the context. Based on the form of axiom, applicable completion rules can be determined. Meanwhile, checking if the conclusion is already in corresponding context’s processed set can be performed. Once a completion rule has been applied, the conclusion axioms and their forms are determined. Once a conclusion is derived, its contexts and whether they are definitely the same as the current context are determined. The conclusion axioms can directly be added into corresponding *scheduled* queues.

5 Evaluation

We implemented our algorithms in our PEL reasoner (written in JAVA). To evaluate its performance we use the Amazon Elastic Computer Cloud (EC2) High-Memory Quadruple Extra Large Instance. It has 8 virtual cores with 3.25 EC2 units each, where each EC2 unit “provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor”. The instance has 60 GB memory allocated to JVM. Our test cases include a real world TBox NotGalen with generated ABox. NotGalen⁻ is extracted from an earlier version of Galen (<http://www.opengalen.org/>) by removing functional role assertions. It contains a moderate-size TBox with 2748 classes, 413 roles and no ABox. To populate the ontology we use the SyGENiA system [15] to generate ABoxes for a small part of the Galen ontology and combine the generated ABoxes of different sizes with the NotGalen⁻ TBox. In this way, we have test ontologies with larger and larger number of individuals, denoted by NGS-1, NGS-5, NGS-10 etc. Such ABoxes are not completely random because, as generated by SyGENiA, they cover axioms that can lead to all possible sources of incompleteness w.r.t. a TBox and certain query (in our evaluation, we query for instances of *PlateletCountProcedure*). Being able to handle such ABoxes means that the reasoner won’t miss any result when dealing with any real-world ABoxes.

For each ontology, we perform ABox materialisation. Note that not all the relations have to be computed in reasoning, but only the necessary ones as indicated in Theorem 1. Nevertheless all relations can be retrieved very easily if needed. Results of our implementation PEL are presented in Table 2. The time shown in our evaluation is the overall computation time. Time unit is second.

Table 2. Results of PEL (in sec) for scalability tests

Ontology	$ \mathcal{IN} $	$ \mathcal{A} $	1 worker	2 workers	4 workers	6 workers
NGS-1	4309	8000	1.378	0.931	0.789	0.733
NGS-5	62639	119852	4.97	4.108	3.746	3.709
NGS-10	211244	437542	16.276	14.196	7.867	6.634
NGS-20	596616	1642664	60.314	52.381	53.559	36.599
NGS-30	865790	3541822	127.517	87.141	83.855	81.964
NGS-40	970925	6036497	180.494	148.802	101.614	105.231
NGS-50	995932	9024356	247.547	204.713	192.055	201.127

From the comparison between different numbers of workers in Table 2 we can see that multiple parallel workers can indeed improve the reasoning performance, even when the ontology contains complex TBox and very large number of individuals. In general, the performance improves when more and more workers are used. With more than 4 workers, the performance may decrease on very large ABoxes. We believe one of the potential reasons is that although the CPU cores can work in parallel, the RAM bandwidth is limited and RAM access is still sequential. In relatively “light-weight” ABox reasoning with large ABox, the RAM access will be enormous and very often so that multiple workers will have to compete for RAM access. This makes memory I/O a potential bottleneck of parallelisation and wastes CPU cycles. Another potential reasoner is that with such a large Java heapsize, the memory management of the JVM will take quite some time. A better management of memory will be an important direction of our future work.

6 Conclusion

In this paper we extended early related work to present a parallel ABox reasoning approach to $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ ontologies. We proposed new completion rules to improve efficiency and scalability of parallel ABox reasoning, and showed that they are complete and sound for ABox reasoning. Our evaluation shows that ABox reasoning can benefit from parallelisation, even for very large ABoxes. In future, We would like to continue the work in two directions. One is to develop distributed algorithms so that the memory cost can also be distributed. The other is to develop target-driven materialisation instead of full materialisation to reduce memory cost.

Acknowledgement

Yuan Ren and Jeff Z. Pan are partially funded by the EU FP7 K-Drive project.

References

1. Aslani, M., Haarslev, V.: Parallel TBox classification in description logics – first experimental results. In: Proceeding of ECAI2010 (2010)
2. Hogan, A., Pan, J.Z., Polleres, A., Decker, S.: SAOR: Template rule optimisations for distributed reasoning over 1 billion linked data triples. In: Proceedings of International Semantic Web Conference. pp. 337–353 (2010)
3. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *J. Web Sem.* 3(2-3), 79–115 (2005)
4. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of \mathcal{EL} ontologies. In: Proceedings of ISWC'11. LNCS, vol. 7032. Springer (2011)
5. Kazakov, Y., Krötzsch, M., Simančík, F.: Practical reasoning with nominals in the \mathcal{EL} family of description logics. In: Proceedings of KR'12 (2012)
6. Liebig, T., Müller, F.: Parallelizing tableaux-based description logic reasoning. In: Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems - Volume Part II. pp. 1135–1144. Springer-Verlag, Berlin, Heidelberg (2007)
7. Maier, R.M.F., Hitzler, P.: A MapReduce algorithm for EL+. In: Proc. of International Workshop of Description Logic (DL2010) (2010)
8. Meissner, A.: Experimental analysis of some computation rules in a simple parallel reasoning system for the \mathcal{ALC} description logic. *Applied Mathematics and Computer Science* 21(1), 83–95 (2011)
9. Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A., van Harmelen, F.: Marvin: Distributed reasoning over large-scale semantic web data. *Web Semant.* 7, 305–316 (2009)
10. Ren, Y., Pan, J.Z., Lee, K.: Parallel ABox reasoning of EL ontologies. In: Proc. of the First Joint International Conference of Semantic Technology (JIST 2011) (2011)
11. Schlicht, A., Stuckenschmidt, H.: Distributed resolution for ALC. In: Description Logics Workshop (2008)
12. Schlicht, A., Stuckenschmidt, H.: Distributed resolution for expressive ontology networks. In: Proceedings of RR'09. pp. 87–101. Springer-Verlag, Berlin, Heidelberg (2009)
13. Serafini, L., Tamilin, A.: Drago: Distributed reasoning architecture for the semantic web. In: ESWC2005. pp. 361–376. Springer (2005)
14. Soma, R., Prasanna, V.K.: Parallel inferencing for owl knowledge bases. In: Proceedings of the 37th International Conference on Parallel Processing. pp. 75–82. ICPP '08, IEEE Computer Society, Washington, DC, USA (2008)
15. Stoilos, G., Cuenca Grau, B., Horrocks, I.: How incomplete is your semantic web reasoner? In: Proc. of AAAI 10. pp. 1431–1436. AAAI Publications (2010)
16. Urbani, J., Kotoulas, S., Maassen, J., Van Harmelen, F., Bal, H.: WebPIE: A web-scale parallel inference engine using MapReduce. *Web Semant.* 10, 59–75 (Jan 2012)
17. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using MapReduce. In: Proceedings of the ISWC '09. LNCS, vol. 5823. Springer (2009)
18. Weaver, J., Hendler, J.A.: Parallel materialization of the finite RDFS closure for hundreds of millions of triples. In: Proc. of ISWC 2009. LNCS, vol. 5823, pp. 682–697. Springer (2009)
19. Wu, G., Qi, G., Du, J.: Finding all justifications of OWL entailments using TMS and MapReduce. In: Proc. of CIKM2011 (2011)

Probabilistic Datalog+/- under the Distribution Semantics

Fabrizio Riguzzi, Elena Bellodi, and Evelina Lamma

ENDIF – University of Ferrara, Via Saragat 1, I-44122, Ferrara, Italy
{fabrizio.riguzzi,elena.bellodi,evelina.lamma}@unife.it

Abstract. We apply the distribution semantics for probabilistic ontologies (named DISPONTE) to the Datalog+/- language. In DISPONTE the formulas of a probabilistic ontology can be annotated with an epistemic or a statistical probability. The epistemic probability represents a degree of confidence in the formula, while the statistical probability considers the populations to which the formula is applied. The probability of a query is defined in terms of finite set of finite explanations for the query, where an explanation is a set of possibly instantiated formulas that is sufficient for entailing the query. The probability of a query is computed from the set of explanations by making them mutually exclusive.

We also compare the DISPONTE approach for Datalog+/- ontologies with that of Probabilistic Datalog+/-, where an ontology is composed of a Datalog+/- theory whose formulas are associated to an assignment of values for the random variables of a companion Markov Logic Network.

1 Introduction

Many authors recognize that representing uncertain information is important for the Semantic Web [18, 12] and recently this was also the topic for a series of workshops [6]. Ontologies are a fundamental component of the Semantic Web and Description Logics (DLs) are often the languages of choice for modeling ontologies. Lately much work has focused on developing tractable DLs, such as the *DL-Lite* family [5], for which answering conjunctive queries is in AC_0 in data complexity.

In a related research direction Cali et al. [3] proposed Datalog+/-, a variant of Datalog for defining ontologies. Datalog+/- is able to express the languages of the *DL-Lite* family [2]. Probabilistic Datalog+/- [9, 8] has been proposed for representing uncertainty in Datalog+/. In this approach an ontology is composed of a Datalog+/- theory and a Markov Logic Network (MLN) [15] and each Datalog+/- formula is associated to an assignment of values to (a subset of) the random variables that are modeled by the MLN. This assignment, called *scenario*, controls the activation of the formulas: they hold only in worlds where the scenario is satisfied.

In the field of logic programming, the distribution semantics [17] has emerged as one of the most effective approaches for integrating logic and probability and underlies many languages such as PRISM [17], ICL [14], Logic Programs with

Annotated Disjunctions [19] and ProbLog [7]. In this semantics the clauses of a probabilistic logic program contain alternative choices annotated with probabilities. Each grounding of a probabilistic clause represents a random variable that can assume a value from the finite set of alternatives. In order to compute the probability of a query, its explanations have to be found, where an explanation is a set of choices that ensure the entailment of the query. The set of explanations must be covering, i.e., it must represent all possible ways of entailing the query. The probability is computed from a covering set of explanations by solving a disjoint sum problem, either using an iterative splitting algorithm [14] or Binary Decision Diagrams [11, 16].

In this paper we apply the distribution semantics to ontological languages and, in particular, to Datalog+/- . We call the approach DISPONTE for “Distribution Semantics for Probabilistic ONTologiEs” (Spanish for “get ready”). The idea is to annotate formulas of a theory with a probability. We consider two types of probabilistic annotation, an epistemic type, that represents a degree of belief in the formula as a whole, and a statistical type, that considers the populations to which the formula is applied. While in the first case the choice is whether to include or not a formula in an explanation, in the latter case the choice is whether to include instantiations of the formula for specific individuals. The probability of a query is again computed from a covering set of explanations by solving the disjoint sum problem.

The paper is organized as follows. Section 2 provides some preliminaries on Datalog+/- . Section 3 presents DISPONTE while Section 4 describes related work. Section 5 concludes the paper.

2 Datalog+/-

Let us assume (i) an infinite set of data constants Δ , (ii) an infinite set of labeled nulls Δ_N (used as “fresh” Skolem terms) and (iii) an infinite set of variables Δ_V . Different constants represent different values (unique name assumption), while different nulls may represent the same value. We assume a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ . We denote by \mathbf{X} vectors of variables X_1, \dots, X_k with $k \geq 0$. A relational schema \mathcal{R} is a finite set of relation names (or predicates). A term t is a constant, null or variable. An atomic formula (or atom) has the form $p(t_1, \dots, t_n)$, where p is an n -ary predicate and t_1, \dots, t_n are terms. A database D for \mathcal{R} is a possibly infinite set of atoms with predicates from \mathcal{R} and arguments from $\Delta \cup \Delta_N$. A conjunctive query (CQ) over \mathcal{R} has the form $q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms having as arguments variables \mathbf{X} and \mathbf{Y} and constants (but no nulls). A Boolean CQ (BCQ) over \mathcal{R} is a CQ having head predicate q of arity 0 (i.e., no variables in \mathbf{X}).

We often write a BCQ omitting the quantifiers. Answers to CQs and BCQs are defined via homomorphisms, which are mappings $\mu : \Delta \cup \Delta_N \cup \Delta_V \rightarrow \Delta \cup \Delta_N \cup \Delta_V$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to term vectors, atoms, sets of atoms, and

conjunctions of atoms. The set of all answers to a CQ $q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$ over a database D , denoted $q(D)$, is the set of all tuples \mathbf{t} over Δ for which there exists a homomorphism $\mu : \mathbf{X} \cup \mathbf{Y} \rightarrow \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The answer to a BCQ q over a database D is Yes, denoted $D \models q$, iff $q(D) \neq \emptyset$.

A *tuple-generating dependency* (or TGD) F is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over \mathcal{R} , called the *body* and the *head* of F , respectively. Such F is satisfied in a database D for \mathcal{R} iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$, there exists an extension h' of h such that $h'(\Psi(\mathbf{X}, \mathbf{Z})) \subseteq D$. We usually omit the universal quantifiers in TGDs. A TGD is *guarded* iff it contains an atom in its body that involves all variables appearing in the body.

Query answering under TGDs is defined as follows. For a set of TGDs T on \mathcal{R} and a database D for \mathcal{R} , the set of models of D given T , denoted $mods(D, T)$, is the set of all (possibly infinite) databases B such that $D \subseteq B$ and every $F \in T$ is satisfied in B . The set of answers to a CQ q on D given T , denoted $ans(q, D, T)$, is the set of all tuples \mathbf{t} such that $\mathbf{t} \in q(B)$ for all $B \in mods(D, T)$. The answer to a BCQ q over D given T is Yes, denoted $D \cup T \models q$, iff $B \models q$ for all $B \in mods(D, T)$.

A Datalog+/- theory may contain also *negative constraints* (or NC), which are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \perp$, where $\Phi(\mathbf{X})$ is a conjunction of atoms (not necessarily guarded). The universal quantifiers are usually left implicit.

Equality-generating dependencies (or EGDs) are the third component of a Datalog+/- theory. An EGD F is a first-order formula of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of F and denoted $body(F)$, is a conjunction of atoms, and X_i and X_j are variables from \mathbf{X} . We call $X_i = X_j$ the *head* of F , denoted $head(F)$. Such F is satisfied in a database D for \mathcal{R} iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. We usually omit the universal quantifiers in EGDs. An EGD F on \mathcal{R} of the form $\Phi(\mathbf{X}) \rightarrow X_i = X_j$ is applicable to a database D for \mathcal{R} iff there exists a homomorphism $\eta : \Phi(\mathbf{X}) \rightarrow D$ such that $\eta(X_i)$ and $\eta(X_j)$ are different and not both constants. If $\eta(X_i)$ and $\eta(X_j)$ are different constants in Δ , then there is a *hard violation* of F . Otherwise, the result of the application of F to D is the database $h(D)$ obtained from D by replacing every occurrence of a non-constant element $e \in \{\eta(X_i), \eta(X_j)\}$ in D by the other element e' (if e and e' are both nulls, then e precedes e' in the lexicographic order).

Example 1. Let us consider the following ontology for a real estate information extraction system, a slight modification of the one presented in [9]:

$$F_1 = ann(X, label), ann(X, price), visible(X) \rightarrow priceElem(X)$$

If X is annotated as a label, as a price, and is visible, then it is a price element.

$$F_2 = ann(X, label), ann(X, priceRange), visible(X) \rightarrow priceElem(X)$$

If X is annotated as a label, as a price range, and is visible, then it is a price element.

$$F_3 = priceElem(E), group(E, X) \rightarrow forSale(X)$$

If E is a price element and is grouped with X , then X is for sale.

$$F_4 = \text{forSale}(X) \rightarrow \exists P \text{price}(X, P)$$

If X is for sale, then there exists a price for X .

$$F_5 = \text{hasCode}(X, C), \text{codeLoc}(C, L) \rightarrow \text{loc}(X, L)$$

If X has postal code C , and C 's location is L , then X 's location is L .

$$F_6 = \text{hasCode}(X, C) \rightarrow \exists L \text{codeLoc}(C, L), \text{loc}(X, L)$$

If X has postal code C , then there exists L such that C has location L and so does X .

$$F_7 = \text{loc}(X, L1), \text{loc}(X, L2) \rightarrow L1 = L2$$

If X has the locations $L1$ and $L2$, then $L1$ and $L2$ are the same.

$$F_8 = \text{loc}(X, L) \rightarrow \text{advertised}(X)$$

If X has a location L then X is advertised.

Suppose we are given the database

$$\begin{aligned} & \text{codeLoc}(\text{ox1}, \text{central}), \text{codeLoc}(\text{ox1}, \text{south}), \text{codeLoc}(\text{ox2}, \text{summertown}) \\ & \text{hasCode}(\text{prop1}, \text{ox2}), \text{ann}(e1, \text{price}), \text{ann}(e1, \text{label}), \text{visible}(e1), \\ & \text{group}(e1, \text{prop1}) \end{aligned}$$

The atomic BCQs $\text{priceElem}(e1)$, $\text{forSale}(\text{prop1})$ and $\text{advertised}(\text{prop1})$ evaluate to true, while the CQ $\text{loc}(\text{prop1}, L)$ has answers $q(L) = \{\text{summertown}\}$. In fact, even if $\text{loc}(\text{prop1}, z_1)$ with $z_1 \in \Delta_N$ is entailed by formula F_5 , formula F_7 imposes that $\text{summertown} = z_1$. If F_7 were absent, then $q(L) = \{\text{summertown}, z_1\}$.

The *chase* is a bottom-up procedure for repairing a database relative to a Datalog+/- theory and can be used for deriving atoms entailed by the database and the theory. If such a theory contains only TGDs, the chase consists of an exhaustive application of the TGD chase rule in a breadth-first fashion. The TGD chase rule consists in adding to the database the head of a TGD if there is an homomorphism between the body and the current database. In order to fill the arguments of the head occupied by existentially quantified variables, “fresh” null values are used.

A BCQ can be answered by performing the chase and checking whether the query is entailed by the extended database that is obtained.

Answering BCQs q over databases, guarded TGDs and NC can be done by, for each constraint $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \perp$, checking that the BCQ $\Phi(\mathbf{X})$ evaluates to false; if one of these checks fails, then the answer to the original BCQ q is positive, otherwise the negative constraints can be simply ignored when answering the original BCQ q .

The chase in the presence of both TGDs and EGDs is computed by iteratively applying (1) a single TGD once and (2) the EGDs, as long as they are applicable (i.e., until a fix point is reached). EGDs are assumed to be separable [4]. Intuitively, separability holds whenever: (i) if there is a hard violation of an EGD in the chase, then there is also one on the database w.r.t. the set of EGDs alone (i.e., without considering the TGDs); and (ii) if there is no hard violation,

then the answers to a BCQ w.r.t. the entire set of dependencies equals those w.r.t. the TGDs alone (i.e., without the EGDs).

A guarded Datalog+/- ontology consists of a database D , a finite set of guarded TGDs T_T , a finite set of negative constraints T_C and a finite set of EGDs T_E that are separable from T_T . The data complexity (i.e., the complexity where both the query and the theory are fixed) of evaluating BCQs relative to a guarded Datalog+/- theory is polynomial [1].

3 The DISPONTE Semantics for Probabilistic Ontologies

A *probabilistic ontology* (D, T) consists of a database D and a set T of *certain formulas*, that take the form of a Datalog+/- TGD, NC or EGD, of *epistemic probabilistic formulas* of the form

$$p_i ::_e F_i \tag{1}$$

where p_i is a real number in $[0, 1]$ and F_i is a TGD, NC or EGD, and of *statistical probabilistic formulas* of the form

$$p_i ::_s F_i \tag{2}$$

where p_i is a real number in $[0, 1]$ and F_i is a TGD.

In formulas of the form (1), p_i is interpreted as an epistemic probability, i.e., as the degree of our belief in formula F_i , while in formulas of the form (2), p_i is interpreted as a statistical probability, i.e., as information regarding random individuals from certain populations. These two types of statements can be related to the work of Halpern [10]: an epistemic statement is a Type 2 statement and a statistical statement is a Type 1 statement.

For example, an epistemic probabilistic concept inclusion TGD of the form

$$p ::_e c(X) \rightarrow d(X) \tag{3}$$

represents the fact that we believe in the truth of $c \subseteq d$, where c and d are interpreted as sets of individuals, with probability p . A statistical probabilistic concept inclusion TGD of the form

$$p ::_s c(X) \rightarrow d(X) \tag{4}$$

instead means that a random individual of class c has probability p of belonging to d , thus representing the statistical information that a fraction p of the individuals of c belongs to d . In this way, the overlap between c and d is quantified. The difference between the two formulas is that, if two individuals belong to class c , the probability that they both belong to d according to (3) is p while according to (4) is $p \times p$.

The idea of DISPONTE is to associate independent Boolean random variables to (instantiations of) the formulas. By assigning values to every random variable we obtain a *world*, the set of logic formulas whose random variable is

assigned to 1. Note that the assumption of independence of the random variables does not limit the set of distributions over the ground logical atoms that can be represented: by possibly introducing extra atoms, any distribution over the atoms that can be represented with a Bayesian network can be represented with a probabilistic ontology.

To clarify what we mean by instantiations, we now define substitutions. Given a formula F , a *substitution* θ is a set of couples X/x where X is a variable universally quantified in the outermost quantifier in F and $x \in \Delta \cup \Delta_N$. The application of θ to F , indicated by $F\theta$, is obtained by replacing X with x in F and by removing X from the external quantification for every couple X/x in θ . An *instantiation* of a formula F is the result of applying a substitution to F .

To obtain a world w of a probabilistic ontology T , we include every certain formula in w . For each axiom of the form (1), we decide whether or not to include it in w . For each axiom of the form (2), we generate all the substitutions for the variables universally quantified in the outermost quantifier and for each instantiation we decide whether or not to include it in w .

There may be an infinite number of instantiations. For each instantiated formula, we decide whether or not to include it in w . In this way we obtain a Datalog+/- theory which can be assigned a semantics as seen in Section 2.

To formally define the semantics of a probabilistic ontology we follow the approach of Poole [14]. An *atomic choice* in this context is a triple (F_i, θ_j, k) where F_i is the i -th formula, θ_j is a substitution and $k \in \{0, 1\}$. If F_i is obtained from a certain formula, then $\theta_j = \emptyset$ and $k = 1$. If F_i is obtained from a formula of the form (1), then $\theta_j = \emptyset$. If F_i is obtained from a formula of the form (2), then θ_j instantiates the variables universally quantified in the outermost quantifier.

A *composite choice* κ is a consistent set of atomic choices, i.e., $(F_i, \theta_j, k) \in \kappa, (F_i, \theta_j, m) \in \kappa \Rightarrow k = m$ (only one decision for each formula). The probability of composite choice κ is $P(\kappa) = \prod_{(F_i, \theta_j, 1) \in \kappa} p_i \prod_{(F_i, \theta_j, 0) \in \kappa} (1 - p_i)$. A *selection* σ is a total composite choice, i.e., it contains one atomic choice (F_i, θ_j, k) for every instantiation $F_i\theta_j$ of formulas of the theory. Since the domain is infinite, every selection is, too. Let us indicate with \mathcal{S}_T the set of all selections. \mathcal{S}_T is infinite as well. A selection σ identifies a theory w_σ called a *world* in this way: $w_\sigma = \{F_i\theta_j | (F_i, \theta_j, 1) \in \sigma\}$. Let us indicate with \mathcal{W}_T the set of all worlds. A composite choice κ identifies a set of worlds $\omega_\kappa = \{w_\sigma | \sigma \in \mathcal{S}_T, \sigma \supseteq \kappa\}$. We define the set of worlds identified by a set of composite choices K as $\omega_K = \bigcup_{\kappa \in K} \omega_\kappa$.

A composite choice κ is an *explanation* for a BCG query q if q is entailed by the database and every world of ω_κ . A set of composite choices K is *covering* with respect to q if every world w_σ in which q is entailed is such that $w_\sigma \in \omega_K$. Two composite choices κ_1 and κ_2 are *incompatible* if their union is inconsistent. A set K of composite choices is *mutually incompatible* if for all $\kappa_1 \in K, \kappa_2 \in K, \kappa_1 \neq \kappa_2 \Rightarrow \kappa_1$ and κ_2 are incompatible.

Explanations can be found by keeping track of the formulas that were used for adding atoms to the database in the chase procedure.

Kolmogorov defined probability functions (or measures) as real-valued functions over an algebra Ω of subsets of a set \mathcal{W} called the *sample space*. The

set Ω is an algebra of \mathcal{W} iff (1) $\mathcal{W} \in \Omega$, (2) Ω is closed under complementation, i.e., $\omega \in \Omega \rightarrow (\mathcal{W} \setminus \omega) \in \Omega$ and (3) Ω is closed under finite union, i.e., $\omega_1 \in \Omega, \omega_2 \in \Omega \rightarrow (\omega_1 \cup \omega_2) \in \Omega$. The elements of Ω are called *measurable sets*. Not every subset of \mathcal{W} need be present in Ω .

Given a sample space \mathcal{W} and an algebra Ω of subsets of \mathcal{W} , a probability measure is a function $\mu : \Omega \rightarrow R$ that satisfies the following axioms: (1) $\mu(\omega) \geq 0$ for all $\omega \in \Omega$, (2) $\mu(\mathcal{W}) = 1$, (3) $\omega_1 \cap \omega_2 = \emptyset \rightarrow \mu(\omega_1 \cup \omega_2) = \mu(\omega_1) + \mu(\omega_2)$ for all $\omega_1 \in \Omega, \omega_2 \in \Omega$.

Poole [14] proposed an algorithm, called *splitting algorithm*, to obtain a set of mutually incompatible K' composite choices from any set of composite choices K such that $\omega_K = \omega_{K'}$. Moreover, he proved that if K_1 and K_2 are both mutually incompatible finite sets of finite composite choices such that $\omega_{K_1} = \omega_{K_2}$ then $\sum_{\kappa \in K_1} P(\kappa) = \sum_{\kappa \in K_2} P(\kappa)$.

These results also hold for the probabilistic ontologies we consider, so we can define a unique probability measure $\mu : \Omega_T \rightarrow [0, 1]$ where Ω_T is defined as the set of sets of worlds identified by finite sets of finite composite choices: $\Omega_T = \{\omega_K | K \text{ is a finite set of finite composite choices}\}$. It is easy to see that Ω_T is an algebra over \mathcal{W}_T .

Then μ is defined by $\mu(\omega_K) = \sum_{\kappa \in K'} P(\kappa)$ where K' is a finite mutually incompatible set of finite composite choices such that $\omega_K = \omega_{K'}$. $\langle \mathcal{W}_T, \Omega_T, \mu \rangle$ is a probability space according to Kolmogorov's definition.

The probability of a BCQ query q is given by $P(q) = \mu(\{w | w \in \mathcal{W}_T \wedge D \cup w \models q\})$. If q has a finite set K of finite explanations such that K is covering then $\{w | w \in \mathcal{W}_T \wedge D \cup w \models q\} \in \Omega_T$ and $P(q)$ is well-defined.

Example 2. Let us consider the following probabilistic ontology, obtained from the one presented in Example 1 by adding probabilistic annotations:

- 0.4 ::_s $F_1 = \text{ann}(X, \text{label}), \text{ann}(X, \text{price}), \text{visible}(X) \rightarrow \text{priceElem}(X)$
- 0.5 ::_s $F_2 = \text{ann}(X, \text{label}), \text{ann}(X, \text{priceRange}), \text{visible}(X) \rightarrow \text{priceElem}(X)$
- 0.6 ::_s $F_3 = \text{priceElem}(E), \text{group}(E, X) \rightarrow \text{forSale}(X)$
- $F_4 = \text{forSale}(X) \rightarrow \exists P \text{price}(X, P)$
- $F_5 = \text{hasCode}(X, C), \text{codeLoc}(C, L) \rightarrow \text{loc}(X, L)$
- $F_6 = \text{hasCode}(X, C) \rightarrow \exists L \text{codeLoc}(C, L), \text{loc}(X, L)$
- 0.8 ::_e $F_7 = \text{loc}(X, L1), \text{loc}(X, L2) \rightarrow L1 = L2$
- 0.7 ::_s $F_8 = \text{loc}(X, L) \rightarrow \text{advertised}(X)$

and the database of Example 1:

```
codeLoc(ox1, central), codeLoc(ox1, south), codeLoc(ox2, summertown),
hasCode(prop1, ox2), ann(e1, price), ann(e1, label), visible(e1),
group(e1, prop1)
```

A covering set of explanations for the query $q = \text{priceElem}(e1)$ is $K = \{\kappa_1\}$ where $\kappa_1 = \{(F_1, \{X/e1\}, 1)\}$. K is also mutually exclusive so $P(q) = 0.4$.

A covering set of explanations for the query $q = forSale(prop1)$ is $K = \{\kappa_1, \kappa_2\}$ where $\kappa_1 = \{(F_1, \{X/prop1\}, 1), (F_3, \{X/prop1\}, 1)\}$ and $\kappa_2 = \{(F_2, \{X/prop1\}, 1), (F_3, \{X/prop1\}, 1)\}$.

An equivalent mutually exclusive set of explanations obtained by applying the splitting algorithm is $K' = \{\kappa'_1, \kappa'_2\}$ where $\kappa'_1 = \{(F_1, \{X/prop1\}, 1), (F_3, \{X/prop1\}, 1), (F_2, \{X/prop1\}, 0)\}$ and $\kappa'_2 = \{(F_2, \{X/prop1\}, 1), (F_3, \{X/prop1\}, 1)\}$ so $P(q) = 0.4 \cdot 0.6 \cdot 0.5 + 0.5 \cdot 0.6 = 0.42$.

A covering set of explanations for the query $q = advertised(prop1)$ is $K = \{\kappa_1, \kappa_2, \kappa_3\}$ with

$$\begin{aligned}\kappa_1 &= \{(F_8, \{X/prop1, L/summertown\}, 1), (F_7, \emptyset, 1)\} \\ \kappa_2 &= \{(F_8, \{X/prop1, L/summertown\}, 1), (F_7, \emptyset, 0)\} \\ \kappa_3 &= \{(F_8, \{X/prop1, L/z_1\}, 1), (F_7, \emptyset, 0)\}\end{aligned}$$

where $z_1 \in \Delta_N$. A mutually exclusive set of explanations is $K' = \{\kappa'_1, \kappa'_2, \kappa'_3\}$ where

$$\begin{aligned}\kappa'_1 &= \{(F_8, \{X/prop1, L/summertown\}, 1), (F_7, \emptyset, 1)\} \\ \kappa'_2 &= \{(F_8, \{X/prop1, L/summertown\}, 1), (F_7, \emptyset, 0), (F_8, \{X/prop1, L/z_1\}, 0)\} \\ \kappa'_3 &= \{(F_8, \{X/prop1, L/z_1\}, 1), (F_7, \emptyset, 0)\}\end{aligned}$$

so $P(q) = 0.7 \cdot 0.8 + 0.7 \cdot 0.2 \cdot 0.3 + 0.7 \cdot 0.2 = 0.742$

Example 3. Let us consider the following ontology, inspired by the **people+pets** ontology proposed in Patel-Schneider et al. [13]:

$$\begin{aligned}0.5 \text{ ::}_s F_1 &= hasAnimal(X, Y), pet(Y) \rightarrow petOwner(X) \\ 0.6 \text{ ::}_s F_2 &= cat(X) \rightarrow pet(X)\end{aligned}$$

and the database $hasAnimal(kevin, fluffy), hasAnimal(kevin, tom), cat(fluffy), cat(tom)$. A covering set of explanations for the query $q = petOwner(kevin)$ is $K = \{\kappa_1, \kappa_2\}$ where $\kappa_1 = \{(F_1, \{X/kevin\}, 1), (F_2, \{X/fluffy\}, 1)\}$ and $\kappa_2 = \{(F_1, \{X/kevin\}, 1), (F_2, \{X/tom\}, 1)\}$. An equivalent mutually exclusive set of explanations is $K' = \{\kappa'_1, \kappa'_2\}$ where:

$$\begin{aligned}\kappa'_1 &= \{(F_1, \{X/kevin\}, 1), (F_2, \{X/fluffy\}, 1), (F_2, \{X/tom\}, 0)\} \\ \kappa'_2 &= \{(F_1, \{X/kevin\}, 1), (F_2, \{X/tom\}, 1)\}\end{aligned}$$

so $P(q) = 0.5 \cdot 0.6 \cdot 0.4 + 0.5 \cdot 0.6 = 0.42$

Example 4. Let us consider the following ontology:

$$\begin{aligned}F_1 &= \exists Y hasAnimal(X, Y), pet(Y) \rightarrow petOwner(X) \\ 0.6 \text{ ::}_s F_2 &= cat(X) \rightarrow pet(X) \\ 0.4 \text{ ::}_e F_3 &= cat(fluffy) \\ 0.3 \text{ ::}_e F_4 &= cat(tom)\end{aligned}$$

and the database $hasAnimal(kevin, fluffy), hasAnimal(kevin, tom)$. A covering set of explanations for the query axiom $q = petOwner(kevin)$ is $K = \{\kappa_1, \kappa_2\}$ where

$$\begin{aligned}\kappa_1 &= \{(F_3, \emptyset, 1), (F_2, \{X/fluuffy\}, 1)\} \\ \kappa_2 &= \{(F_4, \emptyset, 1), (F_2, \{X/tom\}, 1)\}\end{aligned}$$

which, after splitting, becomes $K' = \{\kappa'_1, \kappa'_2, \kappa'_3\}$:

$$\begin{aligned}\kappa'_1 &= \{(F_3, \emptyset, 1), (F_2, \{X/fluuffy\}, 1), (F_4, \emptyset, 1), (F_2, \{X/tom\}, 0)\} \\ \kappa'_2 &= \{(F_3, \emptyset, 1), (F_2, \{X/fluuffy\}, 1), (F_4, \emptyset, 0)\} \\ \kappa'_3 &= \{(F_4, \emptyset, 1), (F_2, \{X/tom\}, 1)\}\end{aligned}$$

so $P(q) = 0.4 \cdot 0.6 \cdot 0.3 \cdot 0.4 + 0.4 \cdot 0.6 \cdot 0.7 + 0.3 \cdot 0.6 = 0.3768$

4 Related Work

Gottlob et al. [9, 8] present probabilistic Datalog+/-, a version of Datalog+/- that allows the representation of probabilistic information by combining Markov Logic Networks with Datalog+/. Each Datalog+/- formula F is annotated with a *probabilistic scenario* λ , an assignment of values to a set of random variables from the MLN associated to the ontology. A full probabilistic scenario assigns a value to all the random variables of the MLN. A probabilistic scenario represents an event that happens when the random variables described by the MLN assume the values indicate in the scenario. Probabilistic formulas then take the form $F : \lambda$.

A probabilistic Datalog+/- is of the form $\Phi = (O, M)$ where O is a set of annotated formulas and M is a MLN. An annotated formula holds when the event associated with its probabilistic annotation holds.

If a is a ground atom, its probability in a probabilistic Datalog+/- ontology $\Phi = (O, M)$, denoted $Pr(a)$, is obtained by summing the probabilities according to M of all full scenarios such that the atom is entailed by the annotated formulas that hold in the scenario.

Example 5. Let us consider the following probabilistic Datalog+/- ontology from [8]:

$$\begin{aligned}F_1 &= visible(X) \rightarrow priceElem(X) : \{ann(X, label), ann(X, price)\} \\ F_2 &= visible(X) \rightarrow priceElem(X) : \{ann(X, label), ann(X, priceRange)\} \\ F_3 &= priceElem(E), group(E, X) \rightarrow forSale(X) : \{sale\} \\ F_4 &= forSale(E) \rightarrow \exists P price(X, P) \\ F_5 &= hasCode(X, C), codeLoc(C, L) \rightarrow loc(X, L) \\ F_6 &= hasCode(X, C) \rightarrow \exists L codeLoc(C, L), loc(X, L) \\ F_7 &= loc(X, L1), loc(X, L2) \rightarrow L1 = L2 : \{uniqueLoc\}\end{aligned}$$

and the MLN

- 0.3 $ann(X, label) \wedge ann(X, price)$
- 0.4 $ann(X, label) \wedge ann(X, priceRange)$
- 0.8 $sale$
- 1.1 $uniqueLoc$

Suppose that this network is grounded with respect to the only constant $e1$. The resulting ground network has 5 Boolean random variables, each corresponding to a logical atom. Therefore, there are 2^5 full scenarios. In this theory $Pr(priceElem(e1)) = 0.492$ and $Pr(forSale(prop1)) = 0.339$.

5 Conclusions

We have presented the application of the distribution semantics for probabilistic ontologies (named DISPONTE) to the Datalog+/- language. DISPONTE is inspired by the distribution semantics of probabilistic logic programming and is a minimal extension of the underlying ontology semantics to allow to represent and reason with uncertain knowledge.

DISPONTE differs from Probabilistic Datalog+/- because the probabilistic interactions among the atoms are modeled directly by means of Datalog+/- formulas rather than by a separate entity. The parameters of DISPONTE Datalog+/- are easier to interpret as they are probabilities (statistical or epistemic) while MLN parameters are weights not directly interpretable as probabilities. Moreover, DISPONTE does not require the prior grounding of the probabilistic atoms, for which the set of constants has to be defined by the user, but allows an on demand grounding on the basis of the terms that are used for inference.

In the future we plan to design inference algorithms for probabilistic Datalog+/- under the DISPONTE semantics.

References

1. Cali, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: International Conference on Principles of Knowledge Representation and Reasoning. pp. 70–80. AAAI Press (2008)
2. Cali, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Symposium on Principles of Database Systems. pp. 77–86. ACM (2009)
3. Cali, A., Gottlob, G., Lukasiewicz, T.: Tractable query answering over ontologies with Datalog+/. In: International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009)
4. Cali, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: IEEE Symposium on Logic in Computer Science. pp. 228–242. IEEE Computer Society (2010)

5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning* 39(3), 385–429 (2007)
6. Costa, P.C.G., d’Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.): Uncertainty Reasoning for the Semantic Web I, ISWC International Workshops, URSW 2005-2007, Revised Selected and Invited Papers, LNCS, vol. 5327. Springer (2008)
7. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: International Joint Conference on Artificial Intelligence. pp. 2462–2467 (2007)
8. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Answering threshold queries in probabilistic Datalog+/- ontologies. In: International Conference on Scalable Uncertainty Management. LNCS, vol. 6929, pp. 401–414. Springer (2011)
9. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Conjunctive query answering in probabilistic Datalog+/- ontologies. In: International Conference on Web Reasoning and Rule Systems. LNCS, vol. 6902, pp. 77–92. Springer (2011)
10. Halpern, J.Y.: An analysis of first-order logics of probability. *Artif. Intell.* 46(3), 311–350 (1990)
11. Kimmig, A., Demoen, B., Raedt, L.D., Costa, V.S., Rocha, R.: On the implementation of the probabilistic logic programming language ProbLog. *Theor. Prac. Log. Prog.* 11(2-3), 235–262 (2011)
12. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. *J. Web Sem.* 6(4), 291–308 (2008)
13. Patel-Schneider, P. F., Horrocks, I., Bechhofer, S.: Tutorial on OWL. In: International Semantic Web Conference (2003), <http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/>
14. Poole, D.: Abducing through negation as failure: stable models within the independent choice logic. *J. Log. Prog.* 44(1-3), 5–35 (2000)
15. Richardson, M., Domingos, P.: Markov logic networks. *Mach. Learn.* 62(1-2), 107–136 (2006)
16. Riguzzi, F.: Extended semantics and inference for the Independent Choice Logic. *J. IGPL* 17(6), 589–629 (2009)
17. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: International Conference on Logic Programming. pp. 715–729. MIT Press (1995)
18. URW3-XG: Uncertainty reasoning for the World Wide Web, final report (2005)
19. Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: International Conference on Logic Programming. LNCS, vol. 3131, pp. 195–209. Springer (2004)

Algebraic Reasoning for \mathcal{SHIQ}

Laleh Roosta Pour and Volker Haarslev

Concordia University, Montreal, Quebec, Canada

Abstract. We present a hybrid tableau calculus for the description logic \mathcal{SHIQ} that decides ABox consistency and uses an algebraic approach for more informed reasoning about qualified number restrictions (QNRs). Benefiting from integer linear programming and several optimization techniques to deal with the interaction of QNRs and inverse roles, our approach provides a more informed calculus. A prototype reasoner based on the hybrid calculus has been implemented that decides concept satisfiability for \mathcal{ALCHIQ} . We provide a set of benchmarks that demonstrate the effectiveness of our hybrid reasoner.

1 Introduction

It is well known that standard tableau calculi for reasoning with qualified number restrictions (QNRs) in description logics (DLs) have no explicit knowledge about set cardinalities implied by QNRs. This lack of information causes significant performance degradations for DL reasoners if the numbers occurring in QNRs are increased. Over the last years a family of hybrid tableau calculi has been developed that address this inefficiency by integrating integer linear programming (ILP) with DL tableau methods, where ILP is used to reason about these set cardinalities. We developed hybrid calculi for the DLs \mathcal{ALCQ} [3], \mathcal{SHQ} [5], and \mathcal{SHOQ} [4, 2]. In this paper we present a new calculus that decides ABox consistency for the DL \mathcal{SHIQ} , which extends \mathcal{SHQ} with inverse roles. This new calculus is a substantial extension of the one for \mathcal{SHQ} [5] since the interaction between inverse roles and QNRs results in back propagation of information possibly adding new back-propagated QNRs, which, in turn, possibly require a conservative extension of solutions obtained by using ILP.

Informally speaking this line of research is based on several principles: (i) role successors are semantically partitioned into disjoint sets such that all members of one set are indistinguishable w.r.t. to their restrictions; (ii) cardinalities of partitions are denoted by non-negative integer variables and the cardinality of a union of partitions can be expressed by the sum of the corresponding partition variables; (iii) all members of a non-empty partition are represented by a proxy element [6] associated with the corresponding partition variable; (iv) cardinality restrictions on a set of role successors imposed by QNRs are encoded as a set of linear inequations; (v) the satisfiability of a set of QNRs is mapped to the problem whether the corresponding system of linear inequations has a non-negative integer solution and the involved proxy elements do not lead to a logical contradiction. This approach is better informed than traditional tableau methods because (i) (implied) set cardinalities are represented using ILP that is independent of the values occurring in QNRs; (ii) the tableau part of the hybrid calculus becomes more efficient because a set of indistinguishable role successors is represented by one proxy; (iii) the partitioning of role successors supports semantic branching on partition cardinalities and more refined dependency-directed backtracking.

2 Preliminaries

In this section we briefly describe syntax and semantics of \mathcal{SHIQ} and its basic inferences services. Furthermore, we define a rewriting that reduces \mathcal{SHIQ} to \mathcal{SHIN}^\setminus in order to apply the atomic decomposition technique [10], which is a basic foundation of our calculus. Let N_C be a set of concept names, N_R a set of role names, $N_{TR} \subseteq N_R$ a set of transitive role names, $N_{SR} \subseteq N_R$ a set of non-transitive role names with $N_{SR} \cap N_{TR} = \emptyset$. The set of roles is defined as $N_{RS} = N_R \cup \{R^- \mid R \in N_R\}$. We define a function Inv such that $Inv(R) = R^-$ if $R \in N_R$, and $Inv(R) = S$ if $R = S^-$. For a set of roles $RO = \{R_1, \dots, R_n\}$, $Inv(RO) = \{Inv(R_1), \dots, Inv(R_n)\}$. A *role hierarchy* \mathcal{R} is a set of axioms of the form $R \sqsubseteq S$ where $R, S \in N_{RS}$ and \sqsubseteq_* is transitive-reflexive closure of \sqsubseteq over $\mathcal{R} \cup \{Inv(R) \sqsubseteq Inv(S) \mid R \sqsubseteq S \in \mathcal{R}\}$. R is called a sub-role of S and S a super-role of R if $R \sqsubseteq_* S$. A role $R \in N_{SR}$ is called simple if R is neither transitive nor has a transitive sub-role. The set of \mathcal{SHIQ} concepts is the smallest set such that (i) every concept name is a concept, and (ii) if A is a concept name, C and D are concepts, R is a role, S is a simple role, $n, m \in \mathbb{N}, n \geq 1$, then $C \sqcap D, C \sqcup D, \neg C, \forall R.C, \exists R.C, \geq nS.C$, and $\leq mS.C$ are also concepts. We consider \top (\perp) as abbreviations for $A \sqcup \neg A$ ($A \sqcap \neg A$). A general concept inclusion axiom (GCI) is an expression of the form $C \sqsubseteq D$ with C, D concepts. A \mathcal{SHIQ} TBox \mathcal{T} w.r.t. a role hierarchy \mathcal{R} is a set of GCIs.

Let I be a set of individual names. A \mathcal{SHIQ} ABox \mathcal{A} w.r.t. a role hierarchy \mathcal{R} is a finite set of assertions of the form of $a : C, \langle a, b \rangle : R$, and $a \neq b$ with $I_A \subseteq I$ the set of individuals occurring in \mathcal{A} and $a, b \in I_A$ and R a role. We assume an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the non-empty set $\Delta^{\mathcal{I}}$ is the domain of \mathcal{I} and $\cdot^{\mathcal{I}}$ is an interpretation function which maps each concept to a subset of $\Delta^{\mathcal{I}}$ and each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Semantics and syntax of the DL \mathcal{SHIQ} are presented in [8].

An interpretation \mathcal{I} holds for a role hierarchy \mathcal{R} iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \sqsubseteq S \in \mathcal{R}$. An interpretation \mathcal{I} satisfies a TBox \mathcal{T} iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every GCI $C \sqsubseteq D \in \mathcal{T}$. An interpretation \mathcal{I} satisfies an ABox \mathcal{A} if it satisfies \mathcal{T} and \mathcal{R} and all assertions in \mathcal{A} such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ if $a : C \in \mathcal{A}$, $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ if $\langle a, b \rangle : R \in \mathcal{A}$, and $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b \in \mathcal{A}$. Such an interpretation is called a model of \mathcal{A} . An ABox \mathcal{A} is consistent iff there exists a model \mathcal{I} of \mathcal{A} . A concept description C is satisfiable iff $C^{\mathcal{I}} \neq \emptyset$.

Let E be a concept expression, then $clos(E)$ defines the usual closure of all concept names occurring in E . Therefore, for a TBox \mathcal{T} if $C \sqsubseteq D \in \mathcal{T}$, then $clos(C) \subseteq clos(D)$ and $clos(D) \subseteq clos(C)$. Likewise for an ABox \mathcal{A} , if $(a : C) \in \mathcal{A}$ then $clos(C) \subseteq clos(\mathcal{A})$. Inspired by [10] we use a satisfiability-preserving rewriting to replace QNRs with unqualified ones. This rewriting uses a new role-set difference operator $\forall(R \setminus R').C$ for which $(\forall(R \setminus R').C)^{\mathcal{I}} = \{x \mid \forall y : \langle x, y \rangle \in R^{\mathcal{I}} \setminus R'^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$. We name the new language \mathcal{SHIN}^\setminus . We have $(\geq nR)^{\mathcal{I}} = (\geq nR.\top)^{\mathcal{I}}$ and $(\leq nR)^{\mathcal{I}} = (\leq nR.\top)^{\mathcal{I}}$. Considering $\neg C$ as the standard negation normal form (NNF) of $\neg C$ we define a recursive function unQ which rewrites \mathcal{SHIQ} concept descriptions and assertions into \mathcal{SHIN}^\setminus .

Definition 1 (unQ). Let R' be a new role in N_R with $\mathcal{R} := \mathcal{R} \cup \{R' \sqsubseteq R\}$ for each transformation. unQ rewrites the axioms as follows: $unQ(C) := C$ if $C \in N_C$, $unQ(\neg C) := \neg C$ if $C \in N_C$ otherwise $unQ(\neg C)$, $unQ(\forall R.C) := \forall R.unQ(C)$, $unQ(C \sqcap D) := unQ(C) \sqcap unQ(D)$, $unQ(C \sqcup D) := unQ(C) \sqcup unQ(D)$,

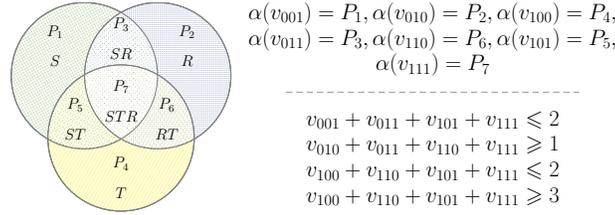


Fig. 1. Atomic decomposition for $\leq 2S \sqcap \geq 1R \sqcap \leq 2T \sqcap \geq 3T$

$unQ(\geq nR.C) := \geq nR' \sqcap \forall R'.unQ(C)$, $unQ(\leq nR.C) := \leq nR' \sqcap \forall (R \setminus R')$.
 $unQ(\neg C)$, $unQ(a : C) := a : unQ(C)$, $unQ(\langle a, b \rangle : R) := \langle a, b \rangle : R$,
 $unQ(a \neq b) := a \neq b$.

Note that this rewriting generates a unique new role for each QNR. For instance, $unQ(\geq nR.C \sqcap \leq mR'.C \sqcap \geq kR''.D)$ is rewritten to $\geq nR_1 \sqcap \forall R_1.C \sqcap \leq mR_2 \sqcap \forall (R \setminus R_2).\neg C \sqcap \geq kR_3 \sqcap \forall R_3.D$ and $\{R_1 \sqsubseteq R, R_2 \sqsubseteq R, R_3 \sqsubseteq R^-\} \subseteq \mathcal{R}$. \mathcal{SHLN}^\wedge is not closed under negation due to the fact that $unQ(\leq nR.C)$ itself creates a negation which must be in NNF before further applying unQ . In order to avoid the whole negating problem for the concept description generated by $unQ(\leq nR.C)$ and $unQ(\geq nR.C)$ the calculus makes sure that the application of unQ starts from the innermost part of an axiom, therefore such concept descriptions will never be negated.

Atomic decomposition A so-called atomic decomposition for reasoning about sets was proposed in [10] and later applied to DLs for reasoning about sets of role fillers (see Def. 3 for role fillers). For instance, for the concept description $\leq 2S \sqcap \geq 1R \sqcap \leq 2T \sqcap \geq 3T$ we get 7 disjoint partitions shown in the left part of Fig. 1, where partition P_1 represents S -fillers that are neither R nor T fillers and P_5 represents fillers in the intersection of S and T that are not R -fillers, etc. For example, due to the disjointness of P_i , $1 \leq i \leq 7$, the cardinality of all S -fillers can be expressed as $|P_1| + |P_3| + |P_5| + |P_7|$ ($|\cdot|$ denotes the cardinality of a set). The satisfiability of the above-mentioned concept descriptions can now be defined by finding a non-negative integer solution for the inequations shown at the bottom-right part of Fig. 1, where v_i denotes the cardinality of P_i with i in unary encoding.

3 SHIQ ABox Calculus

In this section we introduce our calculus and the tableau completion rules for \mathcal{SHIQ} .

Definition 2 (Completion forest). *The algorithm generates a model consisting of a set of arbitrarily connected individuals in I_A as the roots of completion trees. Ignoring the connections between roots, the created model is a forest $\mathcal{F} = (V, E, \mathcal{L}, \mathcal{L}_E, \mathcal{L}_I)$ for a \mathcal{SHIQ} ABox \mathcal{A} . Every node $x \in V$ is labeled by $\mathcal{L}(x) \subseteq \text{clos}(\mathcal{A})$ and $\mathcal{L}_E(x)$ as a set of inequations of the form $\sum_{i \in \mathbb{N}} v_i \bowtie n$ with $n \in \mathbb{N}$ and $\bowtie \in \{\geq, \leq\}$ and variables $v_i \in \mathcal{V}$. Each edge $\langle x, y \rangle \in E$ is labeled by the set $\mathcal{L}(\langle x, y \rangle) \subseteq N_R$. For each node x , $\mathcal{L}_I(x)$ is defined to keep an implied back edge for x equivalent to $\text{Inv}(\mathcal{L}(\langle y, x \rangle))$, where y is a parent of x (see Def. 4). For the nodes with no parents (root nodes) \mathcal{L}_I will be the empty set.*

Definition 3 (-successor, -predecessor, -neighbour, -filler). *Given a completion tree, for nodes x and y with $R \in \mathcal{L}(\langle x, y \rangle)$ and $R \sqsubseteq_* S$, y is called S -successor of x and x is $\text{Inv}(S)$ -predecessor of y . If y is an S -successor or an $\text{Inv}(S)$ -predecessor of x*

then y is called and S -neighbor of x . In addition, if $R \in \mathcal{L}(\langle x, y \rangle)$ then y is an R -filler (role-filler) for x . The R -fillers of x are defined as $Fil(x, R) = \{y \mid \langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \in R^{\mathcal{I}}\}$.

Definition 4 (Precedence). Due to the existence of inverse roles for each pair of individuals x, y , $R \in \mathcal{L}(\langle x, y \rangle)$ imposes $Inv(R) \in \mathcal{L}(\langle y, x \rangle)$. A global counter PR keeps the number of nodes and is increased by one when a new node x is created, setting $PR_x = PR$. Hence, each node is ranked with a PR . A successor of x with the lowest PR is called parent (parent successor) of x and others are called its children. Accordingly, a node x has a lower precedence than a node y if x has a lower rank compared to y . Also, each node has a unique rank and no two nodes have the same rank.

Definition 5 (ξ). Assuming a set of variables \mathcal{V} , a unique variable $v \in \mathcal{V}$ is associated with a set of role names RV_v . Let $\mathcal{V}^R = \{v \in \mathcal{V} \mid R \in RV_v\}$ be the set of all variables which are related to a role R . The function ξ maps number restrictions to inequations such that $\xi(R, \bowtie, n) := (\sum_{v_i \in \mathcal{V}^R} v_i) \bowtie n$.

Definition 6 (Distinct partitions). R_x is defined as the set of related roles for x such that $R_x = \{S \mid \{\xi(S, \geq, n), \xi(S, \leq, m)\} \cap \mathcal{L}_E(x) \neq \emptyset\}$. A partitioning \mathcal{P}_x is defined as $\mathcal{P}_x = \bigcup_{P \subseteq R_x} \{P\} \setminus \{\emptyset\}$. For a partition $P_x \in \mathcal{P}_x$, $P_x^{\mathcal{I}} = (\bigcap_{S \in P_x} Fil^{\mathcal{I}}(x, S)) \setminus (\bigcup_{S \in (R_x \setminus P_x)} Fil^{\mathcal{I}}(x, S))$ with $Fil^{\mathcal{I}}(x, S) = \{y^{\mathcal{I}} \mid y \in Fil(x, S)\}$. Let α be a mapping $\alpha : \mathcal{V} \leftrightarrow \mathcal{P}_x$ for a node x , each variable $v \in \mathcal{V}$ is assigned to a partition $P_x \in \mathcal{P}_x$ such that $\alpha(v) = P_x$.

The definition clearly demonstrates that the fillers of x related to the roles of partition P_x are not the fillers of the roles in $R_x \setminus P$ (other partitions). Therefore, by definition the fillers of x associated with the partitions in \mathcal{P}_x are mutually disjoint w.r.t. the interpretation \mathcal{I} . An arithmetic solution is defined using the function $\sigma : \mathcal{V} \rightarrow \mathbb{N}$ mapping each variable in \mathcal{V} to a non-negative integer. Let \mathcal{V}_x be the set of all variables assigned to a node x such that $\mathcal{V}_x = \{v_i \in \mathcal{V} \mid v_i \in \mathcal{L}_E(x)\}$, a solution Ω for a node x is $\Omega(x) := \{\sigma(v) = n \mid n \in \mathbb{N}, v \in \mathcal{V}_x\}$. The inequation solver uses an objective function to determine whether to minimize the solution or maximize it. We minimize the solution in order to keep the size of the forest small.

The example in Fig. 1 depicts the process of finding an arithmetic solution in more detail. Let $\mathcal{L}(x) = \{\leq 2S, \geq 1R, \leq 2T, \geq 3T\}$ be the label of node x . Applying the atomic decomposition for the related roles $R_x = \{S, R, T\}$ results in seven disjoint partitions such that $\mathcal{P}_x = \{P_i \mid 1 \leq i \leq 7\}$ where $P_1 = \{S\}$, $P_2 = \{R\}$, $P_4 = \{T\}$, $P_3 = \{S, R\}$, $P_5 = \{S, T\}$, $P_6 = \{R, T\}$, $P_7 = \{R, S, T\}$ as shown in Fig. 1. In order to simplify the mapping between variables and partitions, each bit in the binary coding of a variable index represents a specific role in R_x . Therefore, in this example the first bit from right represents S , the second R , and the last T . Since $|R_x| = 3$, the number of variables in \mathcal{V}_x becomes $2^3 - 1$. The mapping of variables and the resulting inequations in $\mathcal{L}_E(x)$ are also shown in Fig. 1.

Definition 7 (Node Cardinality). The cardinality associated with proxy nodes is defined by the mapping $card : \mathcal{V} \rightarrow \mathbb{N}$.

Since the hybrid algorithm requires to have all numerical restrictions encoded as a set of inequations, three functions are defined to map number restrictions (NRs) to inequations and/or further constrain variables. Function ξ (see Def. 5) is used in the \geq -Rule and \leq -Rule as shown in Fig. 2. Function ζ and ς also add new inequations to the label \mathcal{L}_E of a node and modify the variables. The functions ζ and ς are respectively used in the IBE -Rule and $reset_{IBE}$ -Rule as shown in Fig. 2.

Definition 8 (ζ). For a set of roles RO and $k \in \mathbb{N}$, the function $\zeta(RO, k)$ maps number restrictions to inequations via the function ξ for each $R_j \in RO$. $\zeta(RO, k)$ returns a set of inequations such that $\zeta(RO, k) = \{\xi(R_j, \geq, k) \mid R_j \in RO\} \cup \{\xi(R_j, \leq, k) \mid R_j \in RO\}$. For $v \in \mathcal{V}^{R_j}$, if $R_j \in \alpha(v) \wedge \alpha(v) \not\subseteq RO$ then $v \leq 0$ is also returned.

Definition 9 (ς). For a set of roles RO and $k \in \mathbb{N}$, the function $\varsigma(RO, k)$ maps number restrictions to inequations via the function ξ for each $R_j \in RO$. $\varsigma(RO, k)$ returns a set of inequations such that $\varsigma(RO, k) = \{\xi(R_j, \geq, k) \mid R_j \in RO\} \cup \{\xi(R_j, \leq, k) \mid R_j \in RO\}$. For $v \in \mathcal{V}^{R_j}$, if $RO = \alpha(v)$ then $v = k$ is also returned.

Definition 10 (Proper Sub-Role). For each role in existing number restrictions a set will be assigned which contains a specific type of sub-role called proper sub-role. A proper sub-role $\mathfrak{R}(R)$ for a role R is defined as $\mathfrak{R}(R) = \{R_i \mid (R \in N_R \cup Inv(R)) \wedge R_i \sqsubseteq R\}$.

This makes specializing the edges between nodes possible. Therefore, in our algorithm when a role set is assigned to $\mathcal{L}(\langle x, y \rangle)$ a new proper sub-role S_i will be created for each role $S \in \mathcal{L}(\langle x, y \rangle)$, where $\mathfrak{R}(S) = \mathfrak{R}(S) \cup \{S_i\}$, and will be assigned to the edge label. A role in $\mathfrak{R}(S)$ cannot have any proper sub-role. Only roles that occur in number restrictions and their inverses can have proper sub-roles. Since these proper sub-roles do not appear in the logical label of nodes, they do not violate the correctness of our algorithm.

Definition 11 (Blocked Node). Since *SHIQ* does not have the finite model property pair-wise blocking [7] is used. Node y is blocked by node x , also called witness, if $\mathcal{L}(x) = \mathcal{L}(y)$ and for their successors y', x' , $\mathcal{L}(y') = \mathcal{L}(x')$ and $\mathcal{L}(\langle x, x' \rangle) = \mathcal{L}(\langle y, y' \rangle)$. Also, unreachable nodes which were discarded from the forest (due to the application of the *reset-Rule* or *reset_{IBE}-Rule*) are called blocked. In order to detect blocked nodes, all roles in a proper sub-role of role R are considered equivalent to R .

Definition 12 (Clash Triggers). A node x contains a clash if $\{A, \neg A\} \subseteq \mathcal{L}(x)$ (logical clash) or $\mathcal{L}_E(x)$ does not have a non-negative integer solution (arithmetic clash).

The interaction between the tableau rules and the inequation solver is similar to the clash triggers. No particular rule is needed to invoke the inequation solver. For each $\mathcal{L}_E(x)$, there is always a solution (if there exists any) otherwise a clash occurs. If a variable changes or $\mathcal{L}_E(x)$ is extended, a new solution will be calculated automatically. The completion rules for a *SHIQ* ABox are shown in Fig. 2, listed in decreasing priority from top to bottom. Rules in the same cell have the same priority. Rules with lower priorities cannot be applied to a node x , which is not blocked, if there exists any rule with a higher priority still applicable to it. Among the completion rules in Fig. 2, the \sqcap -Rule, \sqcup -Rule, \forall -Rule, \forall_+ -Rule are the same as in standard tableaux. The *merge-Rule*, $\forall \setminus$ -Rule, *ch-Rule*, \geq -Rule, \leq -Rule are similar to [5].

\geq -Rule and \leq -Rule: all number restrictions from $\mathcal{L}(x)$ are collected via these two rules. The function ξ maps them to inequations according to the proper atomic decomposition and adds them to $\mathcal{L}_E(x)$.

IBE-Rule: this rule considers the implied back edge as a set of NRs, maps them to a set of inequations, add the inequations to the \mathcal{L}_E , and determines potential variables that can represent the IBE through elimination of the non-related variables. Assume that for a node x a successor y has been created with $\mathcal{L}(\langle x, y \rangle)$. This implies a back edge for

<i>reset</i> -Rule	<p>if $\{(\leq nR), (\geq nR)\} \cap \mathcal{L}(x) \neq \emptyset$ and $\forall v \in V_x : R \notin \alpha(v)$ then set $\mathcal{L}_E(x) := \emptyset$ and for every successor y of x set $\mathcal{L}(\langle x, y \rangle) := \emptyset$ and, if y in not parent of x set $\mathcal{L}(\langle y, x \rangle) := \emptyset$</p>
<i>reset</i> _{I_{BE}} -Rule	<p>if $Inv(R) \in \mathcal{L}(\langle y, x \rangle)$ but $R \notin \mathcal{L}(\langle x, y \rangle)$ then set $\mathcal{L}_E(x) := \mathcal{L}_E(x) \cup \{\zeta(\mathcal{L}(\langle x, y \rangle), card(y))\}$ and, for every successor y of x set $\mathcal{L}(\langle x, y \rangle) := \emptyset$ and, if y is not parent of x set $\mathcal{L}(\langle y, x \rangle) := \emptyset$</p>
<i>merge</i> -Rule	<p>if there exist root nodes z_a, z_b, z_c for $a, b, c \in I_A$ such that $R' \sqsubseteq_* R_{ab}, R' \in \mathcal{L}(\langle z_a, z_c \rangle)$ then merge the node z_b, z_c and their labels and, replace every occurrence of z_b in the completion graph by z_c</p>
\sqcap -Rule	<p>if $(C_1 \sqcap C_2) \in \mathcal{L}(x)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$</p>
\sqcup -Rule	<p>if $(C_1 \sqcup C_2) \in \mathcal{L}(x)$ and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{X\}$ for some $X \in \{C_1, C_2\}$</p>
\forall -Rule	<p>if $\forall S.C \in \mathcal{L}(x)$ and there is an S-neighbour y of x with $C \notin \mathcal{L}(y)$ then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$</p>
$\forall \setminus$ -Rule	<p>if $\forall R \setminus S.C \in \mathcal{L}(x)$ and there is an R-neighbour y of x with $C \notin \mathcal{L}(y)$ and y is not S-neighbour of x then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$</p>
\forall_+ -Rule	<p>if $\forall S.C \in \mathcal{L}(x)$ and there is some R with $Trans(R)$ and $R \sqsubseteq_* S$ and there is R-neighbour y of x with $\forall R.C \notin \mathcal{L}(y)$ then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$</p>
<i>ch</i> -Rule	<p>if there occurs v in $\mathcal{L}_E(x)$ with $\{v \geq 1, v \leq 0\} \cap \mathcal{L}_E(x) = \emptyset$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{X\}$ for some $X \in \{v \geq 1, v \leq 0\}$</p>
\geq -Rule	<p>if $(\geq nR) \in \mathcal{L}(x)$ and $\xi(R, \geq, n) \notin \mathcal{L}_E(x)$ then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{\xi(R, \geq, n)\}$</p>
\leq -Rule	<p>if $(\leq nR) \in \mathcal{L}(x)$ and $\xi(R, \leq, n) \notin \mathcal{L}_E(x)$ then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{\xi(R, \leq, n)\}$</p>
<i>I_{BE}</i> -Rule	<p>if $\mathcal{L}_I(x) \neq \emptyset$ and $\{\zeta(\mathcal{L}_I(x), 1)\} \cap \mathcal{L}_E(x) = \emptyset$ then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{\zeta(\mathcal{L}_I(x), 1)\}$</p>
<i>BE</i> -Rule	<p>if there exists v occurring in $\mathcal{L}_E(x)$ such that $\sigma(v) = 1, R \in \alpha(v),$ $R \in \mathcal{L}_I(x)$ and y is parent of x with $\mathcal{L}(\langle x, y \rangle) = \emptyset$ then set $\mathcal{L}(\langle x, y \rangle) := \alpha(v)$</p>
<i>RE</i> -Rule	<p>if there exists v occurring in $\mathcal{L}_E(x)$ such that $\sigma(v) = 1, z_a, z_b$ root nodes, $R_{ab} \in \alpha(v)$ with $x, b \in I_A$ and $\mathcal{L}(\langle z_a, z_b \rangle) = \emptyset$ then set $\mathcal{L}(\langle z_a, z_b \rangle) := \alpha(v), \mathcal{L}_I(z_b) := Inv(\alpha(v))$</p>
<i>fil</i> -Rule	<p>if there exists v occurring in $\mathcal{L}_E(x)$ such that $\sigma(v) = n$ with $n > 0,$ x is not blocked and $\neg \exists y : \mathcal{L}(\langle x, y \rangle) = \alpha(v)$ then create a new node y and set $\mathcal{L}(\langle x, y \rangle) := \alpha(v),$ $\mathcal{L}_I(y) := Inv(\alpha(v))$ and $card(y) = n$</p>

Fig. 2. The complete tableaux expansion rules for *SHIQ*-ABox

y with a label $\mathcal{L}_I(y) = \text{Inv}(\mathcal{L}(\langle x, y \rangle))$. This back edge is considered as a set of NRs of the form $\geq 1R_i, \leq 1R_i$ where $R_i \in \text{Inv}(\mathcal{L}(\langle x, y \rangle))$. The *IBE-Rule* transforms the implied back edge into a set of inequations in $\mathcal{L}_E(y)$ of the form $(\sum_{v_j \in \mathcal{V}^{R_i}} v_j) \geq 1$ and $(\sum_{v_j \in \mathcal{V}^{R_i}} v_j) \leq 1$ using the function ζ . Since the inequations representing the back edge are restricted to the value one, only one common variable v_k in these inequations will be $\sigma(v_k) = 1$. In addition, ζ ensures that the potential variables for IBE include all the roles in $\mathcal{L}_I(y)$ (see Def. 9).

reset_{IBE}-Rule: this rule extends \mathcal{L}_E as follows. If for a node y and its parent node x , $\mathcal{L}(\langle x, y \rangle) \neq \text{Inv}(\mathcal{L}(\langle y, x \rangle))$, then it implies that a new role should be considered in $\mathcal{L}_E(x)$ due to the restrictions of its child y . Therefore, the *reset_{IBE}-Rule* fires for x where ζ extends $\mathcal{L}_E(x)$ to consider $\text{Inv}(\mathcal{L}(\langle y, x \rangle))$ and ensures that the specific variable representing it is included in the solution as in Def. 8.

reset-Rule: if a new number restriction with a new role R is added to the logical label of a node x , all its children are discarded from the tree and $\mathcal{L}_E(x) = \emptyset$.

BE-Rule: this rule fills a label of the back edge a node to its parent due to the solution of the inequation solver. If a variable v in a solution exists such that $\sigma(v) = 1$ and $\mathcal{L}_I(y) \subseteq \alpha(v)$, then v represents the back edge and the *BE-Rule* fires and fills the edge label.

We adjust the edges between a pair of nodes to satisfy the nature of the inverse roles between them. Interactions of *IBE-Rule*, *BE-Rule*, and *reset_{IBE}-Rule* maintain this characteristic.

RE-Rule: this rule sets the edge between two root nodes. For nodes $a, b \in I_A$, $(a, b) : R$ is considered as $a : \geq 1R_{ab}, a : \leq 1R_{ab}, b : \geq 1\text{Inv}(R_{ab}), b : \leq 1\text{Inv}(R_{ab})$ with $R_{ab} \sqsubseteq R$ and $\text{Inv}(R_{ab}) \sqsubseteq \text{Inv}(R)$. Therefore, a variable with the value of 1, $\sigma(v) = 1$, for node a that contains R_{ab} represents this edge. The *RE-Rule* fires and fills the edge label.

merge-Rule: the *merge-Rule* merges root nodes. Assume three root nodes $a, b, c \in I_A$ where b, c are respectively R -successor and S -successor of a . These assertions will be translated such that we have $a : \geq 1R_{ab}, a : \leq 1R_{ab}, a : \geq 1S_{ac}, a : \leq 1S_{ac}$ with $R_{ab} \sqsubseteq R$ and $S_{ac} \sqsubseteq S$. If there exists a variable v in an arithmetic solution of node a with $R_{ab}, S_{ac} \in \alpha(v)$, it means that c and b need to be merged. The *merge-Rule* merges b and c and w.l.o.g replaces every occurrence of b with c and all outgoing/incoming edges of b become outgoing/incoming edges of c .

ch-Rule: this rule is necessary to ensure the completeness of the algorithm. The partitions of the atomic decomposition represent all possible combinations of the successors of a particular node. The inequation solver has no knowledge about logical reasons that can force a partition to be empty. Thus, from a semantic branching point of view we need to distinguish between the two cases $v \leq 0$ and $v \geq 1$, where v denotes the cardinality of a partition.

fil-Rule: the *fil-Rule* has the lowest priority among the completion rules. This rule is the only one that generates new nodes, called proxy nodes. For example, if a solution Ω for a node x includes a variable v with $\sigma(v) = 2$, the *fil-Rule* creates a proxy node y with cardinality of 2 and sets the edge label and $\mathcal{L}_I(y)$ as shown in Fig. 2. Since this rule generates proxy nodes based on an arithmetic solution that satisfies all the inequations, there is no need to merge the generated proxy nodes later.

The algorithm preserves role hierarchies in its pre-processing phase. If there occurs a variable $v \in \mathcal{L}_E(x)$, where $R \in \alpha(v)$ and $S \notin \alpha(v)$ and $R \sqsubseteq_* S$, then $v \leq 0$. Therefore, the variables that violate the role hierarchy are set to zero. Due to the space limitations, we refer for the proof to [12].

4 Practical Reasoning

In this section, we discuss the complexity of our algorithm and evaluate its behavior in practical reasoning. Let k be the number of all roles occurring in NRs in a TBox after pre-processing and transforming all QNRs to unqualified ones considering inverse roles. The search space of the hybrid algorithm depends on the number of variables occurring in \mathcal{L}_E labels. Since there are k roles, the number of partitions and their associated variables is bounded by $2^k - 1$. The *ch*-Rule creates two branches for each variable: $v \geq 1$ or $v \leq 0$. Consequently, 2^{2^k} cases could be examined by the inequation solver and the worst-case complexity of the algorithm is double exponential. Moreover, the Simplex method which is used in the hybrid algorithm is NP in the worst case. However, in [11] it is shown that integer programming is in P in the worst case, if the number of variables is bounded. The implemented inequation solver (using Simplex method presented in [1]) minimizes the sum of all variables occurring in the inequations. In addition, we use several optimization techniques and heuristics that can eliminate branches in the search space, therefore, avoiding unnecessary invocations of the *ch*-Rule. These techniques dramatically improve the average complexity of the hybrid algorithm over the worst case of 2^{2^k} .

4.1 Optimization Techniques

In most cases, in order to utilize these theoretical algorithms in practice, optimization techniques are required. Due to the complexity of the algorithm, achieving a good performance may seem infeasible. However, optimization techniques can dramatically decrease the size of the search space by pruning many branches. For instance, a standard technique such as axiom absorption [9] often improves reasoning by reducing the number disjunctions. Here we explain some of the optimization techniques used in our hybrid algorithm.

Variable initialization: the inequation solver starts with the default $v \leq 0$ for all variables and later sets some to $v \geq 1$ to satisfy the inequations according to at-least restrictions. Since the *ch*-Rule is invoked 2^{2^k} times in the worst case to check variables for $v \leq 0$ or $v \geq 1$, default zero setting of variables prevents unnecessary invocations of the *ch*-Rule. Moreover, the boundary value of some of the variables can be determined from the beginning according to the occurrence of the numbers in at-most or/and at-least restrictions. For example, if a variable occurs in an at-least restriction but not in an at-most restriction, then it does not have any arithmetic restriction and is called a *don't care* variable [5]. In addition, the *reset_{I_{BE}}*-Rule specifically determines the value of a single variable and sets some of the rest to zero. Similarly, the *I_{BE}*-Rule sets the value of some variables to zero and enforces some of the others to be the potential choices for the answer, therefore, reducing the solution space. Moreover, in order to avoid unnecessary applications of the *reset_{I_{BE}}*-Rule additional heuristics are applied. For a node x , if a role occurs in an at-least restriction but not in any at-most restriction

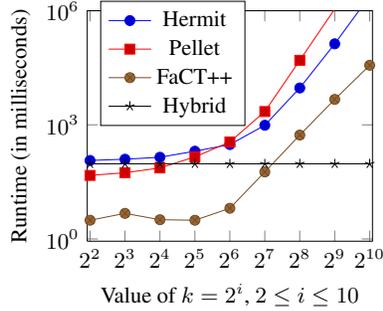


Fig. 3. Runtimes for satisfiable concept $Test_1$ (log-log scale; timeout of 10^6 msec)

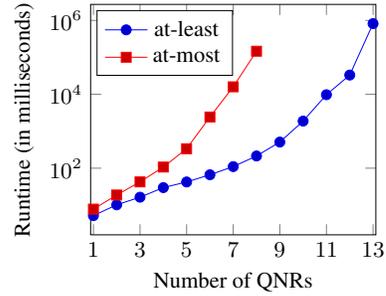


Fig. 4. Runtimes for concept $Test_2$ (using a log scale for the y-axis)

and it is not a sub-role of any role R in a concept of the form $\forall(R \setminus S).C \in \mathcal{L}(x)$, then it cannot be in $\alpha(v)$ where v represents a back edge for node x .

Dependency-directed backtracking: we use backtracking to find sources of logical clashes and then consider the cause of a clash in setting the boundaries of variables in new solutions. This results in pruning branches which would lead to the same clash. If a clash occurs in node x , which was created due to a variable v with $\sigma(v) = k$ and $k \in \mathbb{N}, k \geq 1$, then v must be set to zero. This is called *simple backtracking*. The previous technique can be improved: if a logical clash is encountered due to $\{A, \neg A\} \in \mathcal{L}(x)$, then the source for propagation of these two concepts to $\mathcal{L}(x)$ could be the roles occurring in \forall or $\forall \setminus$ constructs. In this case, the variables which contain all these roles are set to zero. This is called *complex backtracking*. These techniques eliminate many branches in the search space and consequently improving the average complexity of the algorithm.

4.2 A First Evaluation Using Synthetic Benchmarks

In order to evaluate our hybrid algorithm, a prototype reasoner has been implemented in Java using the Web Ontology Language API. The reasoner decides satisfiability of *ALCHIQ* concepts.

We show the performance of the hybrid reasoner by a set of three benchmarks. Fig. 3 demonstrates the performance of different DL reasoners for testing the satisfiability of the concept $Test_1$ defined as $C \sqcap \geq 2kR \sqcap \forall R. (\geq kR^- . C \sqcap \leq kR^- . C)$, where $k = 2^i, 2 \leq i \leq 10$. Fig. 3 compares the reasoning time of our hybrid reasoner with Pellet, FaCT++, and Hermit.¹ All reasoners determine the satisfiability of the concept in less than ~ 100 ms before reaching $k = 2^4$. While our hybrid reasoner maintains its reasoning time (constantly under 100 ms), Pellet, Hermit, and FaCT++ start exhibiting exponential growth in the reasoning time for values higher than $k = 2^4$. For example, for $k = 2^9$ Pellet's reasoning time is more than 18 minutes and for $k = 2^{10}$ it did not finish within the time limit of 1000 seconds. This example demonstrates the independent behavior of our hybrid calculus in the presence of higher values in QNRs interacting with inverse roles.

¹ We used an AMD 3.4GHz quad core CPU with 16 GB of RAM.

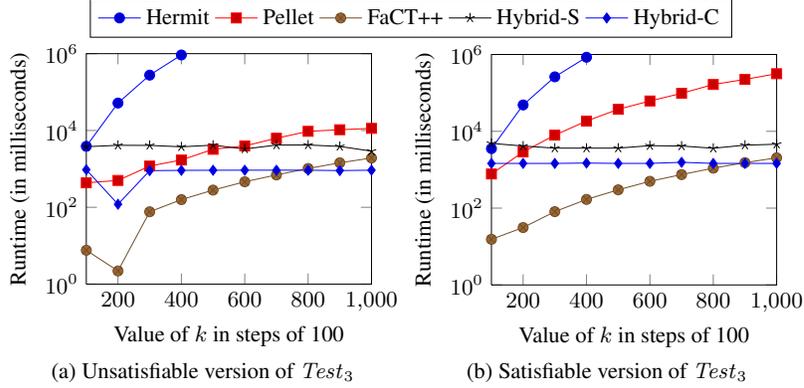


Fig. 5. Linear increase of k (using a log scale for the y-axis)

The second test defines concept $Test_2$ as $\geq 1S.A \sqcap \forall S. \geq 1R.B \sqcap \forall S.\forall R.\forall R^-.P$ with P defined as $\{\sqcap \bowtie 1M_i.C_i \mid 1 \leq i \leq k\}$. Fig. 4 shows the effect of increasing the number of at-least and at-most restrictions in reasoning for $Test_2$. In a model for the concept $Test_2$, the concept expression P is propagated back and will be added to the label of a node which already has $\geq 1R.B$, therefore, we have $(k + 1)$ QNRs. Since for each node which has a parent, an IBE will be considered as a set of two inequations in the \mathcal{L}_E of the node. As shown in Fig. 4, increasing the number of QNRs decreases the performance of the hybrid reasoner. This is due to the fact that a large number of roles in the QNRs increases the number of variables and the size of the search space. Comparing the two diagrams in Fig. 4a and 4b shows that by increasing the number of at-most QNRs the reasoning time for the arithmetic reasoner increases faster than for at-least restrictions. The reason is the heuristic that we explained in Section 4.1. By means of this heuristic, if a role occurs in an at-least restriction and not in any at-most restriction and satisfies the pre-conditions that are mentioned in Section 4.1, then the potential variables for IBE which contain this role are set to zero. Therefore, the number of variables in the search space is decreased.

For the third benchmark we consider the concept $Test_3$ defined as $\geq 8S.A \sqcap \leq 9S.A \sqcap \forall S. (\geq kR.C \sqcap \leq 6T.D \sqcap \geq 5T.D) \sqcap \forall S.\forall R. (\geq 2T.D \sqcap \leq 3T.D) \sqcap \forall S.\forall R.\forall T.\forall T^-. \forall R^-.P$ with $\{R \sqsubseteq M\} \in \mathcal{R}$ and $k \in \{100, \dots, 1000\}$. Fig. 5a displays the runtimes for $Test_3$, where P is defined as $\leq (k - 2)M.(C \sqcup D)$. In this example P is propagated back and causes the unsatisfiability of $Test_3$. Fig. 5b shows the runtimes for $Test_3$ where P is defined as $\leq (k + 1)M.(C \sqcup D)$. In this example P is also propagated back but does not result in the unsatisfiability of $Test_3$. As expected the hybrid reasoner remains stable while the execution times of the other reasoners increase according to the values occurring in the QNRs. As shown in Fig. 5a and 5b, Hermit's behavior is the worst among all the reasoners. Hermit and Pellet show a rapid exponential growth in their reasoning times as a function of k . For $k > 400$, Hermit did not finish within the time limit of 1000 seconds. FaCT++ solves the problem in a more reasonable time, however, it demonstrates its dependency on the value of k as its runtime increases. In addition, Fig. 5 shows the behavior of our hybrid algorithm using *simple* (Hybrid-S)

and *complex* (Hybrid-C) dependency-directed backtracking. The complex backtracking outperforms simple backtracking since it prunes more branches that would lead to the same sort of clash. In addition to improving the performance of the reasoner the optimization techniques used in hybrid reasoner can make its performance more stable.

5 Conclusion and Future Work

The hybrid calculus presented in this paper decides *SHIQ* ABox satisfiability. The implemented prototype demonstrates the improvement on reasoning time for selected benchmarks featuring QNRs and inverse roles. Utilizing algebraic reasoning and applying optimization techniques, the hybrid calculus would be a good solution in case of large numbers occurring in QNRs. In [2] a hybrid algorithm for *SHOQ* using algebraic reasoning has been proposed and extensively analyzed in an empirical evaluation. Due to the nature of nominals, [2] needs to use a global partitioning in contrast to the local partition used in our approach. Several novel techniques to optimize reasoning for *SHOQ* have been designed and evaluated in [2]. Some of these techniques are not exclusively dedicated to nominals and could be applied to our hybrid calculus for *SHIQ* too. For instance, the exponential number of partitions was addressed using a so-called lazy partitioning technique which only generates partitions and their associated variables on demand. We are currently developing a new hybrid prototype for *SHIQ* that integrates suitable optimizations techniques from [2]. We are planning to combine our work on both *SHIQ* and *SHOQ* in order to design a hybrid algebraic calculus for *SHOIQ*.

References

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, 2nd edn. (Sep 2001)
2. Faddoul, J.: Reasoning Algebraically with Description Logics. Ph.D. thesis, Concordia University (September 2011)
3. Faddoul, J., Farsinia, N., Haarslev, V., Möller, R.: A hybrid tableau algorithm for *ALCQ*. In: Proceedings of ECAI 2008, Greece. pp. 725–726 (2008)
4. Faddoul, J., Haarslev, V.: Algebraic tableau reasoning for the description logic *SHOQ*. Journal of Applied Logic 8(4), 334–355 (December 2010)
5. Farsiniamarj, N., Haarslev, V.: Practical reasoning with qualified number restrictions: A hybrid Abox calculus for the description logic *SHQ*. AI Commun. 23, 205–240 (2010)
6. Haarslev, V., Möller, R.: Optimizing reasoning in description logics with qualified number restriction. In: Proc. of DL 2001, USA. pp. 142–151 (Aug 2001)
7. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. Journal of Logic and Computation 9(3), 385–410 (1999)
8. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic *SHIQ*. In: Proc. of CADE-17. pp. 482–496. Springer-Verlag, London, UK (2000)
9. Horrocks, I., Tobies, S.: Reasoning with axioms: Theory and practice. In: In Proc. of KR 2000. pp. 285–296 (2000)
10. Ohlbach, H.J., Koehler, J.: Modal logics, description logics and arithmetic reasoning. Artif. Intell. 109, 1–31 (1999)
11. Papadimitriou, C.H.: On the complexity of integer programming. J. ACM 28, 765–768 (1981)
12. Roosta Pour, L.: A Hybrid ABox calculus using algebraic reasoning for the description logic *SHIQ*. Master’s thesis, Concordia University (January 2012)

Small Datalog Query Rewritings for \mathcal{EL}^*

Giorgio Stefanoni, Boris Motik, and Ian Horrocks

Department of Computer Science, University of Oxford

1 Introduction

Description Logics are a key technology in data management scenarios such as Ontology-Based Data Access (OBDA), a paradigm in which a DL ontology is used to provide a conceptual view of the data [1]. An OBDA system transforms a conjunctive query over the ontology into a query over the data sources [2]. This transformation is independent of the data, so the OBDA approach can thus be used in settings where the data sources provide read-only access to the data, and where the data changes frequently.

Most existing OBDA systems are based on the DL-LITE family of *lightweight* Description Logics [1], which is also the basis for the QL profile of the OWL 2 ontology language. Logics in this family have been designed to allow a conjunctive query posed over the ontology to be rewritten as a first order query over the data sources—that is, queries are first-order rewritable. The query rewriting procedure is independent of the data, and the resulting queries can be evaluated using highly scalable relational database technology. To achieve this, however, the expressive power of DL-LITE is very restrictive. This prevents the OBDA approach from being applied in the life science domain, where many ontologies use DLs from the \mathcal{EL} family [3, 4]. This family provides the basis for the EL profile of OWL2, and many prominent ontologies, such as SNOMED-CT, were developed using this language.

The problem of answering conjunctive queries in \mathcal{EL} has already been studied in the literature, and two orthogonal approaches have been proposed. First, Rosati proposed a pure query rewriting technique which transforms an \mathcal{EL} TBox \mathcal{T} and a conjunctive query q into a DATALOG program $P_{\mathcal{T},q}$ [5]. Second, Lutz et al. introduced a “combined” approach [6, 7]. This technique first materializes certain facts entailed by the ontology in a precomputation step. Then, each user query is rewritten into a polynomial first-order query that, when evaluated over the materialized facts, computes the answers to the user’s query.

Unfortunately, these two approaches exhibit several shortcomings when applied in the context of OBDA. In particular, Rosati’s rewriting technique computes for each user query a fresh DATALOG program whose size depends on both the query and the terminology, which could be very inefficient when dealing with large scale ontologies. The approach by Lutz et al. produces smaller first order rewritings, but the use of materialization means that the technique is only applicable when the data sources provide read/write access to the data; furthermore, materialization can be inefficient if the data changes frequently.

* This work was supported by EPSRC and Alcatel-Lucent.

In this paper, we present a pure query rewriting technique to query answering in \mathcal{EL} . Our approach reinterprets the combined approach proposed by Lutz and colleagues in terms of DATALOG. Our rewriting procedure consists of two distinct steps. The first step rewrites a TBox \mathcal{T} into a DATALOG program $P_{\mathcal{T}}$, whose size depends linearly on the size of \mathcal{T} . Then, at query time, the conjunctive query q is rewritten into a DATALOG query $\langle Q_P, Q_C \rangle$, whose size depends polynomially on q . The two rewriting steps are such that, given an ABox \mathcal{A} , deciding whether $Q_P(a_1, \dots, a_k)$ follows from $P_{\mathcal{T}} \cup Q_C \cup \mathcal{A}$ is equivalent to deciding whether $\langle a_1, \dots, a_k \rangle$ is a certain answer to q over a knowledge base $\langle \mathcal{T}, \mathcal{A} \rangle$.

At last, we summarize our main contributions. First, our rewriting approach, unlike Rosati's, separates the rewriting of the TBox and the query into two distinct steps, thus reducing inefficiency when dealing with large ontologies. Second, our technique does not require the materialization of entailed facts, hence our solution is in the spirit of OBDA and it avoids the problems associated with the materialization of large models. Finally, we set the stage for assessing the utility and the applicability to $P_{\mathcal{T}} \cup Q_C \cup \mathcal{A}$ of optimized DATALOG evaluation techniques, such as *magic sets* and *SLG resolution* [8, 9]. Indeed, heuristic-based evaluation strategies significantly reduce the number of facts needed to answer a query, thus potentially improving the performance of our rewriting approach. In this paper we provide only proof sketches, and refer the reader to [10] for full proofs.

2 Preliminaries

Description Logic \mathcal{EL}

Let N_C, N_R, N_I be pairwise disjoint infinite sets of atomic concepts, atomic roles, and individuals. Together, the sets N_C, N_R , and N_I form the *signature* of an \mathcal{EL} language. Whenever the distinction between atomic concepts and atomic roles is immaterial, we call an element of $N_C \cup N_R$ a *predicate*. The set of \mathcal{EL} *concept expressions* is inductively defined starting from atomic concepts $A \in N_C$ and atomic roles $R \in N_R$ according to the syntax rule: $C \rightarrow A \mid C_1 \sqcap C_2 \mid \exists R.C \mid \top$.

An \mathcal{EL} TBox \mathcal{T} is a finite set of *concept inclusions* of the form $C \sqsubseteq D$; an \mathcal{EL} ABox \mathcal{A} is a finite set of *assertions* of the form $A(a)$ or $R(a, b)$ with a and b individuals; and an \mathcal{EL} *knowledge base* (KB) is a tuple $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where \mathcal{T} is an \mathcal{EL} TBox and \mathcal{A} is an \mathcal{EL} ABox. We denote with $\text{Ind}(\mathcal{A})$ the set of all individuals occurring in the ABox \mathcal{A} . Furthermore, for \mathcal{E} either a TBox or an ABox, $\text{Pred}(\mathcal{E})$ is the set of all predicates occurring in \mathcal{E} .

Semantics is given as usual in terms of first-order interpretations $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a nonempty *domain* and $\cdot^{\mathcal{I}}$ is an *interpretation function*; please refer to [3] for details. In the following, we will extensively use the notion of an *unraveling of an interpretation w.r.t. an ABox*. Consider an interpretation \mathcal{I} and an ABox \mathcal{A} over an arbitrary \mathcal{EL} signature. A *path* p in \mathcal{I} w.r.t. \mathcal{A} is a nonempty finite sequence $c_1 \cdot R_2 \cdot c_2 \cdots R_{n-1} \cdot c_{n-1} \cdot R_n \cdot c_n$ such that $c_1 \in \{a^{\mathcal{I}} \mid a \in \text{Ind}(\mathcal{A})\}$ and for all $1 \leq i \leq n-1$ we have that $\langle c_i, c_{i+1} \rangle \in R_{i+1}^{\mathcal{I}}$ for $R_{i+1} \in N_R$. We say

that a path p has *depth* n and we write $\text{dep}(p) = n$; furthermore, $\text{tail}(p)$ is the last domain element c_n in p . Let $\text{paths}_{\mathcal{A}}(\mathcal{I})$ denote the set of all paths w.r.t. \mathcal{A} occurring in \mathcal{I} . The unraveling \mathcal{J} of \mathcal{I} w.r.t. \mathcal{A} is the following interpretation.

$$\begin{aligned} \Delta^{\mathcal{J}} &= \text{paths}_{\mathcal{A}}(\mathcal{I}) \\ a^{\mathcal{J}} &= a^{\mathcal{I}} \\ A^{\mathcal{J}} &= \{p \in \text{paths}_{\mathcal{A}}(\mathcal{I}) \mid \text{tail}(p) \in A^{\mathcal{I}}\} \\ R^{\mathcal{J}} &= \{\langle a^{\mathcal{J}}, b^{\mathcal{J}} \rangle \mid R(a, b) \in \mathcal{A}\} \cup \{\langle p, p \cdot R \cdot c \rangle \mid \{p, p \cdot R \cdot c\} \subseteq \text{paths}_{\mathcal{A}}(\mathcal{I})\} \end{aligned}$$

In this paper, we deal only with normalized \mathcal{EL} TBoxes. Let A_1 , A , and B be arbitrary concepts from $N_C \cup \{\top\}$. We say that an \mathcal{EL} TBox \mathcal{T} is in *normal form* if each axiom in \mathcal{T} is in one of the following forms: $A \sqsubseteq B$, $A \sqcap A_1 \sqsubseteq B$, $A \sqsubseteq \exists R.B$, or $\exists R.A \sqsubseteq B$. Given an arbitrary \mathcal{EL} TBox \mathcal{T} , we can compute a normalized TBox \mathcal{T}_{norm} of \mathcal{T} in linear time [3].

Querying \mathcal{EL} KBs

Let N_V be an infinite set of *variables* disjoint from N_I . Together, N_V and N_I form the set N_T of *terms*. A *first-order query* q is a first-order formula constructed from the terms in N_T and the predicates from $N_C \cup N_R$ [8]. In general, we write $q = \psi(\vec{x})$ to express that q is the FO formula ψ whose *answer variables* are $\vec{x} = \{x_1, \dots, x_k\}$. A query with k answer variables is a *k-ary query*. A *conjunctive query* (CQ) is a FO query of the form $q = \exists \vec{y}.\psi(\vec{x}, \vec{y})$, where ψ is a conjunction of unary atoms $A(s)$ and binary atoms $R(s, t)$ with s and t terms. The variables \vec{y} are the *quantified variables* of q . In the following, $\text{avar}(q)$ is the set of answer variables of q , and $\text{qvar}(q)$ is the set of quantified variables. Finally, $N_V(q)$ is the set of all variables occurring in q , and $N_T(q)$ is the set of all terms occurring in q . Let $q = \psi(\vec{x})$ be a k -ary FO query with $\vec{x} = \langle x_1, \dots, x_k \rangle$ and let \mathcal{I} be an interpretation. We say that a k -ary tuple of individuals $\langle a_1, \dots, a_k \rangle$ is an answer to q in \mathcal{I} , written $\mathcal{I} \models q[a_1, \dots, a_k]$, if \mathcal{I} satisfies q under the mapping π which sets $\pi(x_i) = a_i$ for all $1 \leq i \leq k$. We call π a *match* for q in \mathcal{I} witnessing $\langle a_1, \dots, a_k \rangle$, written $\mathcal{I} \models^{\pi} q$. We say that $\langle a_1, \dots, a_k \rangle$ is a *certain answer* to q over \mathcal{K} if $\mathcal{I} \models q[a_1, \dots, a_k]$, for all models \mathcal{I} of \mathcal{K} . We denote the set of all certain answers to q over \mathcal{K} with $\text{cert}(q, \mathcal{K})$. Rosati in [5] showed that deciding whether a tuple of individuals is a certain answer to q over \mathcal{K} is PTIME-complete w.r.t. data-complexity (i.e., w.r.t. the size of the ABox); PTIME-complete w.r.t. KB complexity (i.e., w.r.t. the size of \mathcal{K}); and, NP-complete w.r.t. combined complexity (i.e., w.r.t. the size of both \mathcal{K} and q).

Datalog

Let N_B be a nonempty set of *built-in predicates* [11]. Then, a DATALOG rule r is an expression of the form

$$S(\vec{u}) \leftarrow S_1(\vec{u}_1), \dots, S_n(\vec{u}_n), B_{n+1}(\vec{u}_{n+1}) \dots, B_m(\vec{u}_m),$$

where $n, m \geq 0$, $\{S, S_1, \dots, S_n\} \subseteq N_C \cup N_R$, $\{B_{n+1}, \dots, B_m\} \subseteq N_B$, and \vec{u} and \vec{u}_i are tuples of terms of suitable length. A rule is *safe* if each variable occurring in $\vec{u} \cup \vec{u}_{n+1} \cup \dots \cup \vec{u}_m$ also occurs in $\vec{u}_1 \cup \dots \cup \vec{u}_n$. Atom $S(\vec{u})$ is the *head* of the rule, and atoms $S_1(\vec{u}_1), \dots, B_m(\vec{u}_m)$ constitute the *body* of the rule. Whenever the body of a rule r is empty, we call r a *fact*, and we write the rule as $S(\vec{u})$. A DATALOG program P is a set of safe DATALOG rules. Finally, $sch(P)$ is the set of predicates occurring in P .

Next, we define the semantics of a DATALOG program P using *Herbrand interpretations* [8]. The Herbrand Universe of P is the set of all individuals occurring in P . The Herbrand Base of P is the set of all facts that can be constructed from the predicates in $N_C \cup N_R$ and the individuals in the universe of P . A Herbrand interpretation I of P is a subset of the Herbrand Base of P . Note that I does not interpret built-in predicates. As usual, we assume that these predicates are evaluated over a fixed, possibly infinite Herbrand interpretation \mathbf{B} [12]. Then I is a model of P w.r.t. \mathbf{B} if, for all the rules r in P , we have that

$$I \cup \mathbf{B} \models \forall \vec{x} (B_m(\vec{u}_n) \wedge \dots \wedge B_{n+1}(\vec{u}_{n+1}) \wedge S_n(\vec{u}_n) \wedge \dots \wedge S_1(\vec{u}_1) \rightarrow S(\vec{u})),$$

where \vec{x} is a tuple consisting of all variables occurring in the rule. The semantics of a DATALOG program P is defined as the minimal Herbrand interpretation I satisfying P w.r.t. \mathbf{B} , written $\mathbf{M}_{\mathbf{B}}(P)$. Whenever the program does not contain built-in predicates, we do not consider the interpretation \mathbf{B} and we simply write $\mathbf{M}(P)$. The semantics of DATALOG programs can be defined also by means of a fixpoint construction. Then, T_P is the *immediate consequence operator* that maps instances \mathbf{I} over $sch(P)$ to instances over $sch(P)$ as follows. For each rule r in P , if there exists a match π for $S_1(\vec{u}_1) \wedge \dots \wedge S_n(\vec{u}_n) \wedge B_{n+1}(\vec{u}_{n+1}) \wedge \dots \wedge B_m(\vec{u}_m)$ in $\mathbf{I} \cup \mathbf{B}$, then $S(a_1, \dots, a_k)$ is contained in $T_P(\mathbf{I})$ with $a_i = \pi(u_i)$ for each $u_i \in \vec{u}$. One can prove that T_P has a minimum fixpoint T_P^ω such that $T_P^\omega = \mathbf{M}_{\mathbf{B}}(P)$ [8].

Finally, a DATALOG query is a tuple $\langle Q_P, Q_C \rangle$ where Q_P is a predicate symbol and Q_C is a DATALOG program. A tuple of individuals $\langle a_1, \dots, a_k \rangle$ is an *answer* to $\langle Q_P, Q_C \rangle$ over DATALOG program P if $P \cup Q_C \models Q_P(a_1, \dots, a_k)$.

3 Datalog Rewriting for \mathcal{EL} TBoxes

In this section, we show how to transform an \mathcal{EL} TBox \mathcal{T} into a DATALOG program $P_{\mathcal{T}}$ whose size depends linearly on \mathcal{T} . The transformation is such that, for an arbitrary \mathcal{EL} ABox \mathcal{A} , we can use the unraveling of $\mathbf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ to compute the answers to conjunctive queries over $\langle \mathcal{T}, \mathcal{A} \rangle$. Let \mathcal{T} be a TBox over an arbitrary \mathcal{EL} signature. Intuitively, for each axiom α occurring in \mathcal{T} , the program $P_{\mathcal{T}}$ contains a set of DATALOG rules which encode the constraint imposed by α . To achieve this, we have to overcome two issues.

First, \mathcal{EL} concept inclusions of the form $A \sqsubseteq \exists R.B$ require the use of either existential quantifications or Skolem terms in rule heads; however, DATALOG does not allow neither of the two. In order to solve this issue, we use a technique that has been introduced for representing *canonical models* of \mathcal{EL} knowledge bases [3]. That is, for each atomic concept B occurring in \mathcal{T} we introduce a fresh

Axiom α	Set of rules $\Theta(\alpha)$
$A \sqsubseteq B$	$\rightsquigarrow B(X) \leftarrow A(X)$
$A_1 \sqcap A_2 \sqsubseteq B$	$\rightsquigarrow B(X) \leftarrow A_1(X), A_2(X)$
$\exists R.A \sqsubseteq B$	$\rightsquigarrow B(X) \leftarrow R(X, Y), A(Y)$
$A \sqsubseteq \exists R.B$	$\rightsquigarrow R(X, o_B) \leftarrow A(X)$ $B(o_B) \leftarrow A(X)$

Fig. 1. Transformation of \mathcal{EL} Axioms into Rules.

auxiliary individual o_B , which represents the class of existentially quantified individuals of type B . Then, for each axiom of the above form, the program $P_{\mathcal{T}}$ contains the following two rules:

$$R(X, o_B) \leftarrow A(X); \quad B(o_B) \leftarrow A(X).$$

Second, \mathcal{EL} allows for \top to occur in concept expressions. Hence, we need to define in $P_{\mathcal{T}}$ a unary predicate \top , whose extension—given an ABox \mathcal{A} —coincides with the Herbrand universe of $P_{\mathcal{T}} \cup \mathcal{A}$. To achieve this, we restrict our study to a subset of all \mathcal{EL} ABoxes. In particular, we consider only those ABoxes \mathcal{A} such that $\text{Pred}(\mathcal{A}) \subseteq \text{Pred}(\mathcal{T})$. That is, each predicate occurring in the ABox \mathcal{A} must occur also in the TBox \mathcal{T} . Then, in our DATALOG program, for each atomic concept A and for each atomic role R occurring in \mathcal{T} , we add the following rules:

$$\top(X) \leftarrow A(X); \quad \top(X) \leftarrow R(X, Y); \quad \top(Y) \leftarrow R(X, Y).$$

This is only one of the several ways in which we can encode such a predicate. In fact, another possibility would be—as suggested by Rosati in [5]—to assume that each ABox \mathcal{A} contains an assertion $\top(a)$ for each individual $a \in \text{Ind}(\mathcal{A})$. We believe that in the context of OBDA—where the focus is to provide access to arbitrary data-sources—it is important to make as few assumptions as possible on the physical realization of the ABox. For this reason, we prefer the option presented above.

Next, we formalize the transformation of a TBox \mathcal{T} into a DATALOG program $P_{\mathcal{T}}$. Let $\mathbf{Aux} = \{o_A \mid A \in N_C\} \cup \{o_{\top}\}$ be a set of *auxiliary individuals* distinct from N_I . Then, the program $P_{\mathcal{T}}$ is constructed from terms in $N_{\mathcal{T}} \cup \mathbf{Aux}$ and predicates in $N_C \cup N_R \cup \{\top\}$ as follows. The transformation uses the function Θ , shown in Figure 1, to transform each axiom in the (normalized) TBox \mathcal{T} into a set of DATALOG rules. The DATALOG program $P_{\mathcal{T}}$ is then defined as follows.

$$P_{\mathcal{T}} = \bigcup_{\alpha \in \mathcal{T}} \Theta(\alpha) \cup_{A \in \text{Pred}(\mathcal{T}) \cap N_C} \top(X) \leftarrow A(X) \cup_{R \in \text{Pred}(\mathcal{T}) \cap N_R} \top(X) \leftarrow R(X, Y), \top(Y) \leftarrow R(X, Y)$$

The following result readily follows from the definition of the program.

Proposition 1. *For an arbitrary \mathcal{EL} TBox \mathcal{T} , DATALOG program $P_{\mathcal{T}}$ can be computed in time linear in the size of \mathcal{T} .*

Consider an arbitrary \mathcal{EL} ABox \mathcal{A} . Next, we prove that the unraveling \mathcal{U} w.r.t. \mathcal{A} of $M(P_{\mathcal{T}} \cup \mathcal{A})$ can be used to answer conjunctive queries over $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. We do so in two distinct steps. First, we introduce the notion of chase of an \mathcal{EL} knowledge base \mathcal{K} . Second, we show that the chase of \mathcal{K} is isomorphic to \mathcal{U} .

The *chase* of an \mathcal{EL} knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, written $\text{chase}(\mathcal{K})$, is a possibly infinite Herbrand interpretation defined inductively by starting from \mathcal{A} and then *applying* axioms occurring in the TBox to assertions occurring in the ABox. In our definition of the chase, we use function terms to denote existentially quantified individuals. Hence, the definition of ABox assertion is extended in a natural way to accommodate for assertions over function terms. We denote with u and w terms that can be either individuals or function terms. Next, we define an operator $\Gamma_{\mathcal{T}}$ that chases an ABox by applying the axioms occurring in the TBox \mathcal{T} . In the definition, we use assertions of the form $\top(u)$ to assert that u is a member of the \mathcal{EL} *concept expression* \top . For \mathcal{S} an arbitrary ABox, $\Gamma_{\mathcal{T}}(\mathcal{S})$ is the smallest ABox containing \mathcal{S} and closed under the following *chasing rules*.

- (cr1) If $\{A(u)\} \subseteq \mathcal{S}$ and $A \sqsubseteq B \in \mathcal{T}$, then $\{B(u)\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S})$.
- (cr2) If $\{A_1(u), A_2(u)\} \subseteq \mathcal{S}$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$, then $\{B(u)\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S})$.
- (cr3) If $\{R(u, w), A(w)\} \subseteq \mathcal{S}$ and $\exists R.A \sqsubseteq B \in \mathcal{T}$, then $\{B(u)\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S})$.
- (cr4) If $\{A(u)\} \subseteq \mathcal{S}$ and $A \sqsubseteq \exists R.B \in \mathcal{T}$, then

$$\{R(u, f(u, R, B)), B(f(u, R, B))\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S}).$$

- (cr5) If u occurs in \mathcal{S} , then $\{\top(u)\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S})$.

We now define an infinite sequence of finite ABoxes \mathcal{A}_i for $i \in \mathbb{N}$.

$$\begin{aligned} \mathcal{A}_0 &= \mathcal{A} \\ \mathcal{A}_{i+1} &= \Gamma_{\mathcal{T}}(\mathcal{A}_i) \end{aligned}$$

Finally, the chase of \mathcal{K} is the infinite union of all such ABoxes \mathcal{A}_i .

$$\text{chase}(\mathcal{K}) = \bigcup_{i \in \mathbb{N}} \mathcal{A}_i$$

It is clear that our construction of the chase of \mathcal{K} is *fair*. In fact, for each $i \in \mathbb{N}$ we have that \mathcal{A}_{i+1} is the result of exhaustively applying—to all possible assertions occurring in \mathcal{A}_i —all applicable axioms in \mathcal{T} . At last, we want to point out that $\text{chase}(\mathcal{K})$ can be used to compute the certain answers to a CQ q over \mathcal{K} .

Proposition 2 ([5]). *Let \mathcal{K} be an \mathcal{EL} knowledge base. Further, let q be a k -ary conjunctive query. Then, for each k -ary tuple of individuals $\langle a_1, \dots, a_k \rangle$, we have*

$$\langle a_1, \dots, a_k \rangle \in \text{cert}(q, \mathcal{K}) \text{ if and only if } \text{chase}(\mathcal{K}) \models q[a_1, \dots, a_k].$$

So, by proving that the unraveling \mathcal{U} of $M(P_{\mathcal{T}} \cup \mathcal{A})$ is isomorphic to $\text{chase}(\mathcal{K})$, we establish that \mathcal{U} can be used to answer conjunctive queries over \mathcal{K} . To prove the structural equivalence of \mathcal{U} and $\text{chase}(\mathcal{K})$, we define a function h mapping

paths occurring in \mathcal{U} to terms in $\text{chase}(\mathcal{K})$. We define h by induction on the depth of paths occurring in \mathcal{U} as follows.

BASE CASE. Consider an arbitrary $p \in \Delta^{\mathcal{U}}$ with $\text{dep}(p) = 1$. We set $h(p) := p$.

INDUCTIVE STEP. Let $p = t_1 \cdot R_2 \cdot t_2 \cdots t_{n-1} \cdot R_n \cdot t_n$ be a path occurring in \mathcal{U} such that $h(p)$ has not been defined yet, but $h(t_1 \cdots R_{n-1} \cdot t_{n-1}) = u$. We distinguish between two cases depending on the type of the individual t_n .

1. If t_n occurs in the ABox, we set $h(p) := t_n$.
2. If t_n is of the form o_B , we set $h(p) := f(u, R_n, B)$.

Theorem 1 shows that h is an isomorphism between the two structures. Intuitively, for the only-if direction, we show that h is an injective homomorphism from \mathcal{U} to $\text{chase}(\mathcal{K})$ by induction on the depth of paths occurring in \mathcal{U} ; for the if-direction, by induction on the construction of $\text{chase}(\mathcal{K})$ we prove that h is a surjective function and that it is a homomorphism from $\text{chase}(\mathcal{K})$ to \mathcal{U} .

Theorem 1. *Function h is an isomorphism from \mathcal{U} to $\text{chase}(\mathcal{K})$.*

Since the unraveling of $\mathbf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is generally infinite, this result alone does not provide us with an algorithm for answering queries in \mathcal{EL} . In the next section, we show how to rewrite a user query q into a DATALOG query $\langle Q_P, Q_C \rangle$ such that $P_{\mathcal{T}} \cup \mathcal{A} \cup Q_C \models Q_P(a_1, \dots, a_k)$ if and only if $\langle a_1, \dots, a_k \rangle \in \text{cert}(q, \mathcal{K})$ and thus solve the problem.

4 Polynomial Query Rewriting in Datalog

In the previous section, we have seen that for an arbitrary \mathcal{EL} KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ evaluating a conjunctive query q over the unraveling of the Herbrand model of $P_{\mathcal{T}} \cup \mathcal{A}$ is equivalent to computing the certain answers to q over \mathcal{K} . In this section, we develop a query rewriting procedure that reduces the computation of $\text{cert}(q, \mathcal{K})$ to the problem of evaluating a suitably constructed DATALOG query over $P_{\mathcal{T}} \cup \mathcal{A}$. We achieve this in two steps. First, we present an interesting property of a certain class of interpretations. Second, we show how this result can be used to develop a query rewriting procedure in our DATALOG setting.

We use the notions of \mathcal{A} -connected and split interpretations from [6, 13]. Let \mathcal{I} be an interpretation and let \mathcal{A} be an ABox over an arbitrary \mathcal{EL} signature. We say that \mathcal{I} is \mathcal{A} -connected if, for each domain element $c \in \Delta^{\mathcal{I}}$, there exists a path $p \in \text{paths}_{\mathcal{A}}(\mathcal{I})$ such that $\text{tail}(p) = c$. Furthermore, \mathcal{I} is a *split* interpretation if, for all domain elements $c, c' \in \Delta^{\mathcal{I}}$, we have that $c \notin \{a^{\mathcal{I}} \mid a \in \text{Ind}(\mathcal{A})\}$ and $\langle c, c' \rangle \in R^{\mathcal{I}}$ imply $c' \notin \{a^{\mathcal{I}} \mid a \in \text{Ind}(\mathcal{A})\}$. Intuitively, in an \mathcal{A} -connected interpretation \mathcal{I} , for each domain element c_n it is always possible to find a path $a^{\mathcal{I}} \cdot R_2 \cdot c_2 \cdots R_n \cdot c_n$ such that $a \in \text{Ind}(\mathcal{A})$. Furthermore, if \mathcal{I} is split, then each domain element that is not the image of an individual can be related by a role only with elements that themselves do not interpret individuals.

Then, let \mathcal{J} be the unraveling w.r.t. \mathcal{A} of a split and \mathcal{A} -connected interpretation \mathcal{I} and let q be a conjunctive query. Lutz et al. in [6, 13] showed that it is possible to reduce the problem of answering q in \mathcal{J} to evaluating a first-order query rewriting q^* of q over \mathcal{I} . Roughly speaking, the query rewriting q^* rules

out some spurious answers for q in \mathcal{I} that cannot be reproduced in \mathcal{J} . More specifically, we have to ensure that the answer variables of q , the variables of q mapped to cyclic portions of \mathcal{I} , and the variables of q mapped to nontree portions of \mathcal{I} are all matched only to the domain elements in $\{a^{\mathcal{I}} \mid a \in \text{Ind}(\mathcal{A})\}$.

We now briefly outline how we can construct such an FO rewriting q^* for q [6]. Let \sim_q be the smallest equivalence relation over $N_T(q)$ that is closed under the following rule: if $R(s, t)$ and $R(s', t')$ occur in q and $t \sim_q t'$, then $s \sim_q s'$. Then, for each equivalence class ζ of \sim_q , we let $t_\zeta \in \zeta$ be an arbitrary, but fixed, representative of the class. Also, for each such equivalence class ζ and for each atomic role R occurring in q , we let $\text{Pred}(\zeta, R)$ be the following set.

$$\text{Pred}(\zeta, R) = \{t \in N_T(q) \mid R(t, t') \text{ occurs in } q \text{ with } t' \in \zeta\}$$

Next, we define three sets of terms that correspond to the above mentioned cases.

- $\text{Fork}_=$ is the set of all pairs $\langle \text{Pred}(\zeta, R), t_\zeta \rangle$ such that ζ is an equivalence class of \sim_q and $|\text{Pred}(\zeta, R)| > 1$.
- Fork_\neq is the set of all quantified variables $v \in \text{qvar}(q)$ for which atoms $R(s, v)$ and $S(s', t)$ exist in q such that $R \neq S$ and $v \sim_q t$.
- Cyc is the set of all variables $v \in \text{qvar}(q)$ for which atoms

$$R_0(t_0, t'_0), \dots, R_m(t_m, t'_m), \dots, R_n(t_n, t'_n)$$

exist in q such that $m, n \geq 0$; for some $i \leq n$ we have that $t_i \sim_q v$; for each $j < n$ we have that $t'_j \sim_q t_{j+1}$; and $t'_n \sim_q t_m$.

We are now ready to formally specify the FO query rewriting q^* . In the definition, we assume that Aux is a fresh predicate not occurring in q and \mathcal{K} and that every interpretation \mathcal{I} interprets Aux as $\Delta^{\mathcal{I}} \setminus \{a^{\mathcal{I}} \mid a \in \text{Ind}(\mathcal{A})\}$. Then, formulae q_1 and q_2 are defined as follows.

$$q_1 = \bigwedge_{v \in \text{avar}(q) \cup \text{Fork}_\neq \cup \text{Cyc}} \neg \text{Aux}(v)$$

$$q_2 = \bigwedge_{\langle \text{Pred}(\zeta, R), t_\zeta \rangle \in \text{Fork}_=} \neg \text{Aux}(t_\zeta) \vee \bigwedge_{t, t' \in \text{Pred}(\zeta, R)} (t = t')$$

Finally, we set $q^* = q \wedge q_1 \wedge q_2$. It turns out that q^* can be computed in polynomial time w.r.t. q [6]. In the same paper, Lutz et al. prove the following result.

Proposition 3. *Let \mathcal{A} be an arbitrary \mathcal{EL} ABox, let \mathcal{I} be a split and \mathcal{A} -connected interpretation, and let \mathcal{J} be the unraveling of \mathcal{I} w.r.t. \mathcal{A} . Then, for every k -tuple of individuals $\langle a_1, \dots, a_k \rangle$, we have that*

$$\mathcal{I} \models q^*[a_1 \dots, a_k] \text{ if and only if } \mathcal{J} \models q[a_1 \dots, a_k].$$

This result applies to our DATALOG rewriting of \mathcal{EL} TBoxes. Indeed, for an arbitrary \mathcal{EL} KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we have that $\text{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is a split and \mathcal{A} -connected interpretation. The intuition behind the argument is as follows. We show that $\text{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is split by noticing that rules encoded in $P_{\mathcal{T}}$ do not allow for the derivation of facts of the form $R(o_B, a)$ for $a \in \text{Ind}(\mathcal{A})$ and $o_B \in \mathbf{Aux}$. To see that $\text{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is \mathcal{A} -connected, we just recall that $\text{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is minimal and, hence, all the derived facts must be “grounded” w.r.t. the facts in \mathcal{A} .

Theorem 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an \mathcal{EL} knowledge base. Then, $\mathbf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is a split and \mathcal{A} -connected interpretation.*

Hence, for an arbitrary k -ary CQ q and for each k -tuple of individuals $\langle a_1, \dots, a_k \rangle$, we have that $\mathbf{M}(P_{\mathcal{T}} \cup \mathcal{A}) \models q^*[a_1 \dots, a_k]$ if and only if $\langle a_1, \dots, a_k \rangle \in \text{cert}(q, \mathcal{K})$.

Note that q^* is a first-order query, and we are unaware of systems capable of evaluating first-order queries over DATALOG programs. Therefore, we next show how to transform q^* into a DATALOG query $\langle Q_P, Q_C \rangle$ such that $\langle a_1, \dots, a_k \rangle \in \text{cert}(q, \mathcal{K})$ if and only if $P_{\mathcal{T}} \cup \mathcal{A} \cup Q_C \models Q_P(a_1 \dots, a_k)$. We construct such a DATALOG query $\langle Q_P, Q_C \rangle$ by applying to q^* a simplified version of the Lloyd-Topor transformation [14, 15].

Definition 1 (Datalog Rewriting). *Let $q(\vec{x})$ be a k -ary CQ whose quantified variables are among \vec{y} ; let Cyc , Fork_{\neq} , and $\text{Fork}_{=}$ be as specified above; let $\langle \text{Pred}(\zeta^1, R^1), t_{\zeta}^1 \rangle, \dots, \langle \text{Pred}(\zeta^n, R^n), t_{\zeta}^n \rangle$ be an arbitrary enumeration of $\text{Fork}_{=}$; let p_0, p_1, \dots, p_n be fresh predicates; and let Named be a built-in with a pre-terminated, possibly infinite Herbrand interpretation $\mathbf{N} = \{\text{Named}(a) \mid a \in N_I\}$. Query Q_C then contains the following safe DATALOG rules:*

$$p_0(\vec{x}, \vec{y}) \leftarrow q, \quad \bigwedge_{v \in \text{avar}(q) \cup \text{Fork}_{\neq} \cup \text{Cyc}} \text{Named}(v) \quad (1)$$

$$p_i(\vec{x}, \vec{y}) \leftarrow p_{i-1}(\vec{x}, \vec{y}), \text{Named}(t_{\zeta}^i) \quad \text{for } 1 \leq i \leq n \quad (2)$$

$$p_i(\vec{x}, \vec{y}) \leftarrow p_{i-1}(\vec{x}, \vec{y}), \quad \bigwedge_{t, t' \in \text{Pred}(\zeta^i, R^i)} t = t' \quad \text{for } 1 \leq i \leq n \quad (3)$$

$$Q_P(\vec{x}) \leftarrow p_n(\vec{x}, \vec{y}) \quad (4)$$

One may think that the recursive definition of predicates p_i for $1 \leq i \leq n$ could be simplified by writing $Q_P(\vec{x}) \leftarrow p_0(\vec{x}, \vec{y}) \dots p_n(\vec{x}, \vec{y})$ and by defining each p_i as:

$$p_i(\vec{x}, \vec{y}) \leftarrow \text{Named}(t_{\zeta}^i) \quad p_i(\vec{x}, \vec{y}) \leftarrow \bigwedge_{t, t' \in \text{Pred}(\zeta^i, R^i)} t = t'$$

Unfortunately, these rules are not safe. Safe rules, on the one hand, provide us with a clear and unambiguous semantics. On the other hand, unsafe rules are also computationally more expensive for bottom-up computation, since each variable in the head may be bound to an arbitrary individual in the universe of the program. For this reason, we prefer our, slightly more involved, solution.

Proposition 4. *For an arbitrary k -ary conjunctive query q , query $\langle Q_P, Q_C \rangle$ can be computed in polynomial time w.r.t. the size of q .*

Proof. We note that \sim_q can be computed in polynomial time w.r.t. the size of q [6] and, therefore, also the sets Cyc , Fork_{\neq} , and $\text{Fork}_{=}$ can be computed in polynomial time w.r.t. q . Furthermore, the size of the body of rule $p_0(\vec{x}, \vec{y})$ depends linearly on the size of q , Cyc , and Fork_{\neq} . Also, for each pair $\langle \text{Pred}(\zeta, R), t_{\zeta} \rangle$ in $\text{Fork}_{=}$, the program Q_C contains exactly two rules. The size of these two rules depends linearly on the size of $\langle \text{Pred}(\zeta, R), t_{\zeta} \rangle$. Thus, we conclude that $\langle Q_P, Q_C \rangle$ can be computed in polynomial time with respect to the size of q . \square

In [10], we prove that our rewriting is correct—that is, that answering $\langle Q_P, Q_C \rangle$ over $P_{\mathcal{T}} \cup \mathcal{A}$ is equivalent to computing $\text{cert}(q, \mathcal{T}, \mathcal{A})$. This follows from Proposition 3 and the fact that our DATALOG query is the result of transforming the FO rewriting q^* along the lines of the Lloyd-Topor transformation.

Theorem 3. *Let \mathcal{K} be an \mathcal{EL} knowledge base and let q be a k -ary CQ over \mathcal{K} . Then, for every k -tuple of individuals $\langle a_1, \dots, a_k \rangle$, we have that*

$$\langle a_1, \dots, a_k \rangle \in \text{cert}(q, \mathcal{K}) \text{ if and only if } P_{\mathcal{T}} \cup \mathcal{A} \cup Q_C \models Q_P(a_1, \dots, a_k).$$

Finally, we investigate the complexity of our rewriting procedure.

Theorem 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an \mathcal{EL} KB, let q be a k -ary CQ, and let $\langle a_1, \dots, a_k \rangle$ be a tuple of individuals. We can decide $P_{\mathcal{T}} \cup \mathcal{A} \cup Q_C \models Q_P(a_1, \dots, a_k)$ in polynomial time w.r.t. the size of \mathcal{K} and in non-deterministic polynomial time with respect to the size of both \mathcal{K} and q .*

Proof. We have already argued that the size of DATALOG program $P_{\mathcal{T}}$ depends linearly on the size of the TBox \mathcal{T} and that the DATALOG rewriting $\langle Q_P, Q_C \rangle$ can be computed in PTIME w.r.t. q . Also, we note that the arity of predicates and the number of variables occurring in $P_{\mathcal{T}} \cup \mathcal{A} \cup Q_C$ do not depend on \mathcal{K} . Finally, from an implementation point-of-view (as suggested in [12]), the built-in predicate **Named** can be considered as an assertion in the ABox \mathcal{A} with a different physical realization: it is not directly stored in the ABox but it is implemented as a procedure which is evaluated during the execution of the program. Clearly, such a procedure can be implemented to run in time polynomial in \mathcal{K} . It follows that we can compute the minimal Herbrand model of $P_{\mathcal{T}} \cup \mathcal{A} \cup Q_C$ in time polynomial in the size of \mathcal{K} [8]. The membership in NP follows directly from the considerations above and from the fact that we can guess and check in nondeterministic polynomial time a match π for Q_P in $M(P_{\mathcal{T}} \cup \mathcal{A} \cup Q_C)$. \square

5 Conclusions

In this paper, we introduce a query rewriting approach to query answering in \mathcal{EL} . In our approach, the process of computing the certain answers to a CQ q over an \mathcal{EL} KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is divided into two distinct steps. A first preprocessing step in which the TBox \mathcal{T} is transformed into a DATALOG program $P_{\mathcal{T}}$, whose size is linear in \mathcal{T} . Then, at query time, the query q is independently rewritten into a DATALOG query $\langle Q_P, Q_C \rangle$, whose size is polynomial in q . Finally, computing $\text{cert}(q, \mathcal{K})$ amounts to evaluating the DATALOG query $\langle Q_P, Q_C \rangle$ over $P_{\mathcal{T}} \cup \mathcal{A}$.

In future, we plan to extend our approach to deal with $\mathcal{ELH}_{\perp}^{dr}$. Lutz et al. have already proposed a combined approach for query answering in this DL [13]. However, differently from their solution, we would like the DATALOG rewriting $\langle Q_P, Q_C \rangle$ to be independent from the role inclusions contained in the TBox. Additionally, we plan to extend our work to cover nominals, which raises the question on how to efficiently handle equality in DATALOG [2].

References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Rosati, R.: Ontologies and databases: the DL-Lite approach. In Tessaris, S., Franconi, E., eds.: *Semantic Technologies for Informations Systems – 5th Int. Reasoning Web Summer School (RW 2009)*. Volume 5689. (2009) 255–356
2. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. *J. Applied Logic* **8**(2) (2010) 186–209
3. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In Kaelbling, L.P., Saffiotti, A., eds.: *IJCAI, Professional Book Center* (2005) 364–369
4. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope further. In Clark, K., Patel-Schneider, P.F., eds.: *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*. (2008)
5. Rosati, R.: On conjunctive query answering in \mathcal{EL} . In Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.Y., Tessaris, S., eds.: *Description Logics*. Volume 250 of *CEUR Workshop Proceedings.*, CEUR-WS.org (2007)
6. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in \mathcal{EL} using a database system. In: *Proceedings of the OWLED 2008 Workshop on OWL: Experiences and Directions*. (2008)
7. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to ontology-based data access. In: *IJCAI, AAAI Press* (2011)
8. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of databases*. Addison-Wesley (1995)
9. Chen, W., Warren, D.S.: Query evaluation under the well-founded semantics. In: *Proceedings of the twelfth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. PODS '93, New York, NY, USA, ACM (1993) 168–179
10. Stefanoni, G., Motik, B., Horrocks, I.: Small datalog query rewriting for \mathcal{EL} . Technical report, University of Oxford (2012) Available at <http://www.cs.ox.ac.uk/people/giorgio.stefanoni/pubs/2012/tr/dl2012.pdf>.
11. Ullman, J.D.: *Principles of database and knowledge-base systems*, Vol. I. Computer Science Press, Inc., New York, NY, USA (1988)
12. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng.* **1**(1) (1989) 146–166
13. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic \mathcal{EL} using a relational database system. In Boutilier, C., ed.: *IJCAI*. (2009) 2070–2075
14. Lloyd, J., Topor, R.: Making prolog more expressive. *The Journal of Logic Programming* **1**(3) (1984) 225 – 240
15. Decker, S.: *Semantic web methods for knowledge managment*. PhD thesis, University of Karlsruhe (2002)

Extended Caching and Backjumping for Expressive Description Logics

Andreas Steigmiller¹, Thorsten Liebig², and Birte Glimm¹

¹ Ulm University, Ulm, Germany, <first name>.<last name>@uni-ulm.de

² derivo GmbH, Ulm, Germany, liebig@derivo.de

1 Motivation

Due to the wide range of modelling constructs supported by the expressive DL *SROIQ*, the typically used tableau algorithms in competitive reasoning systems such as FaCT++ [16], Hermit³, or Pellet [14] have a very high worst-case complexity. The development of tableau optimisations that help to achieve practical efficiency is, therefore, a long-standing challenge in DL research (see, e.g., [11, 17]). A very effective and widely implemented optimisation technique is “caching”, where one caches, for a set of concepts, whether they are known to be, or can safely be assumed to be, satisfiable or unsatisfiable [4]. If the set of concepts appears again in a model abstraction, then a cache-lookup allows for skipping further applications of tableau rules. Unfortunately, with increasing expressivity naively caching become unsound, for instance, due to the possible interaction of inverse roles with universal restrictions [1, Chapter 9].

With this contribution we push the boundary of the caching optimisation to the expressive DL *SROIQ*. The developed unsatisfiability caching method is based on a sophisticated dependency management, which further enables better informed tableau backtracking and more efficient pruning (Section 3). Our techniques are grounded in the widely implemented tableau calculus for *SROIQ* [9], which makes it easy to transfer our results into existing implementations. The optimisations are integrated within a novel reasoning system, called Konclude [13]. Our empirical evaluation shows that the proposed optimisations result in significant performance improvements (Section 4).

2 Preliminaries

Model construction calculi, such as tableau, decide the consistency of a knowledge base \mathcal{K} by trying to construct an abstraction of a model for \mathcal{K} , a so-called “completion graph”. A completion graph G is a tuple $(V, E, \mathcal{L}, \neq)$, where each node $x \in V$ represents one or more individuals, and is labelled with a set of concepts, $\mathcal{L}(x)$, which the individuals represented by x are instances of; each edge $\langle x, y \rangle$ represents one or more pairs of individuals, and is labelled with a set of roles, $\mathcal{L}(\langle x, y \rangle)$, which the pairs of individuals represented by $\langle x, y \rangle$ are instances of. The relation \neq records inequalities, which must hold between nodes, e.g., due to at-least cardinality restrictions.

³ <http://www.hermit-reasoner.com>

The algorithm works by initialising the graph with one node for each Abox individual/nominal in the input KB, and using a set of expansion rules to syntactically decompose concepts in node labels. Each such rule application can add new concepts to node labels and/or new nodes and edges to the completion graph, thereby explicating the structure of a model. The rules are repeatedly applied until either the graph is fully expanded (no more rules are applicable), in which case the graph can be used to construct a model that is a *witness* to the consistency of \mathcal{K} , or an obvious contradiction (called a *clash*) is discovered (e.g., both C and $\neg C$ in a node label), proving that the completion graph does not correspond to a model. The input knowledge base \mathcal{K} is *consistent* if the rules (some of which are non-deterministic) can be applied such that they build a fully expanded, clash free completion graph. A cycle detection technique called *blocking* ensures the termination of the algorithm.

2.1 Dependency Tracking

Dependency tracking keeps track of all dependencies that cause the existence of concepts in node labels, roles in edge labels as well as accompanying constraints such as inequalities that must hold between nodes. Dependencies are associated with so-called *facts*, defined as follows:

Definition 1 (Fact) We say that G contains a concept fact $C(x)$ if $x \in V$ and $C \in \mathcal{L}(x)$, G contains a role fact $r(x, y)$ if $\langle x, y \rangle \in E$ and $r \in \mathcal{L}(\langle x, y \rangle)$, and G contains an inequality fact $x \dot{\neq} y$ if $x, y \in V$ and $(x, y) \in \dot{\neq}$. We denote the set of all (concept, role, or inequality) facts in G as Facts_G .

Dependencies now relate facts in a completion graph to the facts that caused their existence. Additionally, we annotate these relations with a running index, called dependency number, and a branching tag to track non-deterministic expansions:

Definition 2 (Dependency) Let d be a pair in $\text{Facts}_G \times \text{Facts}_G$. A dependency is of the form $d^{n,b}$ with $n \in \mathbf{N}_0$ a dependency number and $b \in \mathbf{N}_0$ a branching tag.

We inductively define the dependencies Dep_G for G : If G is an initial completion graph, then $\text{Dep}_G = \emptyset$. We initialise the beginning for the next dependency numbers n_m with 1 if $\text{Dep}_G = \emptyset$; otherwise, $n_m = 1 + \max\{n \mid d^{n,b} \in \text{Dep}_G\}$. Let R be a tableau rule applicable to a completion graph G . If R is non-deterministic, the next non-deterministic branching tag b_R for R is $1 + \max\{0\} \cup \{b \mid d^{n,b} \in \text{Dep}_G\}$; for R deterministic, $b_R = 0$. Let G' be the completion graph obtained from G by applying R with c_0, \dots, c_k the facts to satisfy the preconditions of R and c'_0, \dots, c'_ℓ the newly added facts in G' , then

$$\begin{aligned} \text{Dep}_{G'} &= \text{Dep}_G \cup \{(c'_j, c_i)^{n,b} \mid 0 \leq i \leq k, 0 \leq j \leq \ell, n = n_m + (j * k) + i, \\ &\quad b = \max\{b_R\} \cup \{b' \mid (c_i, c')^{n',b'} \in \text{Dep}_G\}\}. \end{aligned}$$

The branching tag indicates which facts were added non-deterministically:

Definition 3 (Non-deterministic Dependency) For $d^{n,b} \in \text{Dep}_G$ with $d = (c_1, c_2)$, let $D_d = \{(c_2, c_3)^{n',b'} \mid (c_2, c_3)^{n',b'} \in \text{Dep}_G\}$. The dependency $d^{n,b}$ is a non-deterministic dependency in G if $b > 0$ and either $D_d = \emptyset$ or $\max\{b' \mid (c, c')^{n',b'} \in D_d\} < b$.

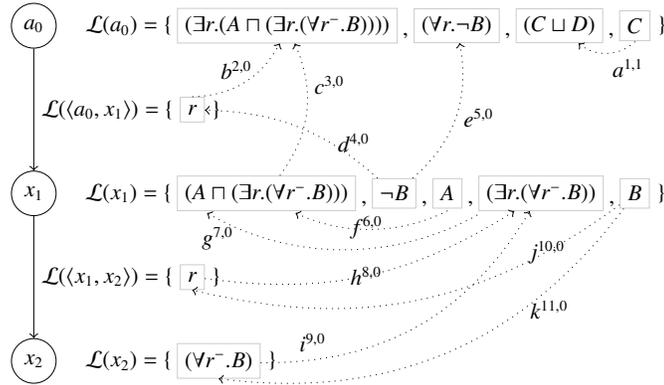


Fig. 1. Tracked dependencies for all facts in the generated completion graph

Figure 1 illustrates a completion graph obtained in the course of testing the consistency of a knowledge base with three concept assertions:

$$a_0 : (\exists r.(A \sqcap (\exists r.(\forall r^-.B)))) \quad a_0 : (\forall r.^-B) \quad a_0 : (C \sqcup D).$$

Thus, the completion graph is initialised with the node a_0 , which has the three concepts in its label. Initially, the set of dependencies is empty. For the concepts and roles added by the application of tableau rules, the dependencies are shown with dotted lines, labelled with the dependency. The dependency number increases with every new dependency. The branching tag is only non-zero for the non-deterministic addition of C to the label of a_0 in order to satisfy the disjunction $(C \sqcup D)$. Note the presence of a clash due to B and $\neg B$ in the label of x_1 .

3 Extended Caching and Backtracking

In the following we introduce improvements to caching and backjumping by presenting a more informed dependency directed backtracking strategy that also allows for extracting precise unsatisfiability cache entries.

3.1 Dependency Directed Backtracking

Dependency directed backtracking is an optimisation that can effectively prune irrelevant alternatives of non-deterministic branching decisions. If branching points are not involved in clashes, it will not be necessary to compute any more alternatives of these branching points, because the other alternatives cannot eliminate the cause of the clash. To identify involved non-deterministic branching points, all facts in a completion graph are labelled with information about the branching points they depend on. Thus, the united information of all clashed facts can be used to identify involved branching points. A typical realisation of dependency directed backtracking is backjumping [1, 17], where the dependent branching points are collected in the dependency sets for all facts.

3.2 Unsatisfiability Caching

Another widely used technique to increase the performance of a tableau implementation is caching. For *unsatisfiability caching*, one caches sets of concepts that are known to be unsatisfiable. For such a cache entry, it holds that any *superset* is also unsatisfiable. Thus, if, in a future tableau expansion, one encounters a node label that is a superset of a cache entry, one can stop expanding the branch.

Analogously to unsatisfiability caching, one can define satisfiability caching and many systems combine both caches. We focus here, however, on unsatisfiability caching since the two problems are quite different in nature and the required data structures for an efficient cache retrieval can differ significantly. Before we define how and when we create cache entries, we formalise our notion of an unsatisfiability cache.

Definition 4 (Unsatisfiability Cache) *Let \mathcal{K} be a knowledge base and $\text{Con}_{\mathcal{K}}$ the set of (sub-)concepts that occur in \mathcal{K} . An unsatisfiability cache $\text{UC}_{\mathcal{K}}$ for \mathcal{K} is a subset of $2^{\text{Con}_{\mathcal{K}}}$ such that each cache entry $S \in \text{UC}_{\mathcal{K}}$ is unsatisfiable w.r.t. \mathcal{K} . An unsatisfiability retrieval for $\text{UC}_{\mathcal{K}}$ and a completion graph G for \mathcal{K} takes a set of concepts $S \subseteq \text{Con}_{\mathcal{K}}$ from a node label of G as input. If $\text{UC}_{\mathcal{K}}$ contains a set $S_{\perp} \subseteq S$, then S_{\perp} is returned; otherwise, the empty set is returned.*

Although node labels can have many concepts that are not involved in any clashes, most implementations cache complete node labels. One reason for this might be that the often used backjumping [1, 17] only allows the identification of all branching points involved in a clash, but there is no information about how the clash is exactly caused. We refer to this form of caching as *label caching*. Systems that use label caching typically also only check whether the exact node label from the current tableau is in the cache.

The creation of cache entries rapidly becomes difficult with increasing expressivity of the used DL. Already with blocking for the DL \mathcal{ALC} , one can easily generate invalid cache entries [6]. Apart from a node x with a clash in its label, the question is which other node labels are also unsatisfiable. For \mathcal{ALC} , this is the case for all labels from x up to the ancestor y with the last non-deterministic expansion. With \mathcal{ALCI} , a non-deterministic rule application on a descendant node of x can be involved in the clash, which makes it difficult to determine node labels that can be cached. Nevertheless, caching techniques for \mathcal{ALCI} have been proposed [2, 3, 5], but the difficulty further increases in the presence of nominals and, to the best of our knowledge, the problem of caching with inverses and nominals has not yet been addressed in the literature. In order to avoid unsound results, current systems often deactivate caching in presence of inverse roles or at least in presence of nominals, especially with combined satisfiability and unsatisfiability caching [14, 17].

The extraction of a small still unsatisfiable subset of a node label would yield better cache entries. The use of subset retrieval methods for the cache further increases the number of cache hits. We call such a technique *precise caching*. Although techniques to realise efficient subset retrieval exist [8], unsatisfiability caches that use such subset retrieval are only implemented in very few DL reasoners [7].

Going back to the example in Figure 1, for the node x_1 the set $\{\neg B, (\exists r.(\forall r^{-}.B))\}$ could be inserted into the cache as well as $\{\neg B, (A \sqcap (\exists r.(\forall r^{-}.B)))\}$. The number of cache

entries should, however, be kept small, because the performance of the retrieval decreases with an increasing number of entries. Thus, the insertion of concepts for which the rule application is cheap (e.g., concept conjunction) should be avoided. Concepts that require the application of non-deterministic or generating rules are more suitable, because the extra effort of querying the unsatisfiability cache before the rule application can be worth the effort. Optimising cache retrievals for incremental changes further helps to efficiently handle multiple retrievals for the same node with identical or slightly extended concept labels.

The creation of new unsatisfiability cache entries based on dependency tracking can be done during backtracing, which is also coupled with the dependency directed backtracking as described next.

3.3 Dependency Backtracing

The dependency tracking defined in Section 2.1 completely retains all necessary information to exactly trace back the cause of the clash. Thus, this *backtracing* is qualified to identify all involved non-deterministic branching points for the dependency directed backtracking and also to identify small unsatisfiable sets of concepts that can be used to create new unsatisfiability cache entries.

Algorithm 1 performs the backtracing of facts and their tracked dependencies in the presence of inverse roles and nominals. If all facts and their dependencies are collected on the same node while backtracing, an unsatisfiability cache entry with these facts can be generated, assuming all facts are concept facts. As long as no nominal or Abox individual occurs in the backtracing, the unsatisfiability cache entries can also be generated while all concept facts have the same node depth. Thus, an important task of the backtracing algorithm is to hold as many facts as possible within the same node depth to allow for the generation of many cache entries. To realise the backtracing, we introduce the following data structure:

Definition 5 (Fact Dependency Node Tuple) *A fact dependency node tuple for G is a triple $\langle c, d^{n,b}, x \rangle$ with $c \in \text{Facts}_G$, $d^{n,b} \in \text{Dep}_G$ and $x \in V$. As abbreviation we also write $\langle C, d^{n,b}, x \rangle$ if c is the concept fact $C(x)$.*

If a clash is discovered in the completion graph, a set of fact dependency node tuples is generated for the backtracing. Each tuple consists of a fact involved in the clash, an associated dependency and the node where the clash occurred. The algorithm gets this set T of tuples as input and incrementally traces the facts back from the node with the clash to nodes with depth 0 (Abox individuals or root nodes).

In each loop round (line 3) some tuples of T are exchanged with tuples, whose facts are the cause of the exchanged one. To identify which tuple has to be traced back first, the current minimum node depth (line 4) and the maximum branching tag (line 5) are extracted from the tuples of T . All tuples, whose facts are located on a deeper node and whose dependencies are deterministic, are collected in the set A . Such tuples will be directly traced back until their facts reach the current minimum node depth (line 10-12). If there are no more tuples on deeper nodes with deterministic dependencies, i.e., $A = \emptyset$, the remaining tuples from deeper nodes with non-deterministic dependencies

Algorithm 1 Backtracing Algorithm

Require: A set of fact dependency node tuples T obtained from clashes

```
1: procedure DEPENDENCYBACKTRACING( $T$ )
2:    $pendingUnsatCaching \leftarrow false$ 
3:   loop
4:      $min_D \leftarrow \text{MINIMUMNODEDEPTH}(T)$ 
5:      $max_B \leftarrow \text{MAXIMUMBRANCHINGTAG}(T)$ 
6:      $A \leftarrow \{t \in T \mid \text{NODEDEPTH}(t) > min_D \wedge \text{HASETERMINISTICDEPENDENCY}(t)\}$ 
7:      $C \leftarrow \emptyset$ 
8:     if  $A \neq \emptyset$  then
9:        $pendingUnsatCaching \leftarrow true$ 
10:      for all  $t \in A$  do
11:         $T \leftarrow (T \setminus t) \cup \text{GETCAUSETUPLESBYDEPENDENCY}(t)$ 
12:      end for
13:    else
14:       $B \leftarrow \{t \in T \mid \text{NODEDEPTH}(t) > min_D \wedge \text{BRANCHINGTAG}(t) = max_B\}$ 
15:      if  $B = \emptyset$  then
16:        if  $pendingUnsatCaching = true$  then
17:           $pendingUnsatCaching \leftarrow \text{TRYCREATEUNSATCACHEENTRY}(T)$ 
18:        end if
19:        if  $\text{HASNODEPENDENCY}(t)$  for all  $t \in T$  then
20:           $pendingUnsatCaching \leftarrow \text{TRYCREATEUNSATCACHEENTRY}(T)$ 
21:        return
22:        end if
23:         $C \leftarrow \{t \in T \mid \text{BRANCHINGTAG}(t) = max_B\}$ 
24:      end if
25:       $t \leftarrow \text{ANYELEMENT}(B \cup C)$ 
26:      if  $\text{HASETERMINISTICDEPENDENCY}(t)$  then
27:         $T \leftarrow (T \setminus t) \cup \text{GETCAUSETUPLESBYDEPENDENCY}(t)$ 
28:      else
29:         $b \leftarrow \text{GETNONDETERMINISTICBRANCHINGPOINT}(t)$ 
30:        if  $\text{ALLALTERNATIVESOFNONDETBRANCHINGPOINTPROCESSED}(b)$  then
31:           $T \leftarrow T \cup \text{LOADTUPLESFROMNONDETBRANCHINGPOINT}(b)$ 
32:           $T \leftarrow (T \setminus t) \cup \text{GETCAUSETUPLESBYDEPENDENCY}(t)$ 
33:           $T \leftarrow \text{BACKTRACETUPLESBEFOREBRANCHINGPOINT}(T, b)$ 
34:           $pendingUnsatCaching \leftarrow \text{TRYCREATEUNSATCACHEENTRY}(T)$ 
35:        else
36:           $T \leftarrow \text{BACKTRACETUPLESBEFOREBRANCHINGPOINT}(T, b)$ 
37:           $\text{SAVETUPLESNONDETBRANCHINGPOINT}(T, b)$ 
38:           $\text{JUMPBACKTO}(max_B)$ 
39:        return
40:        end if
41:      end if
42:    end if
43:  end loop
44: end procedure
```

and the current branching tag are copied into B (line 14) in the next round. If B is not empty, one of these tuples (line 25) and the corresponding non-deterministic branching point (line 29) are processed. The backtracing is only continued, if all alternatives of the branching point are computed as unsatisfiable. In this case, all tuples, saved from the backtracing of other unsatisfiable alternatives, are added to T (line 31). Moreover, for c the concept fact in the tuple t , t can be replaced with tuples for the fact on which c non-deterministically depends (line 32).

For a possible unsatisfiability cache entry all remaining tuples, which also depend on the non-deterministic branching point, have to be traced back until there are no tuples with facts of some alternatives of this branching point left (line 33). An unsatisfiability cache entry is only generated (line 34), if all facts in T are concept facts for the same node or on the same node depth.

Unprocessed alternatives of a non-deterministic branching point have to be computed before the backtracing can be continued. It is, therefore, ensured that tuples do not consist of facts and dependencies from this alternative, which also allows for releasing memory (line 36). The tuples are saved to the branching point (line 37) and the algorithm jumps back to an unprocessed alternative (line 38).

If B is also empty, but there are still dependencies to previous facts, some tuples based on the current branching tag have to remain on the current minimum node depth. These tuples are collected in the set C (line 23) and are processed separately one per loop round, similar to the tuples of B , because the minimum node depth or maximum branching tag may change. The tuples of C can have deterministic dependencies, which are processed like the tuples of A (line 27). If all tuples have no more dependencies to previous facts, the algorithm terminates (line 21).

Besides the creation of unsatisfiability cache entries after non-deterministic dependencies (line 34), cache entries may also be generated when switching from a deeper node to the current minimum node depth in the backtracing (line 9 and 17) or when the backtracing finishes (line 20). The function that tries to create new unsatisfiability cache entries (line 17, 20, and 34) returns a Boolean flag that indicates whether the attempt has failed, so that the attempt can be repeated later.

For an example, we consider the clash $\{\neg B, B\}$ in the completion graph of Figure 1. The initial set of tuples for the backtracing is T_1 (see Figure 2). Thus, the minimum node depth for T_1 is 1 and the maximum branching tag is 0. Because there are no tuples on a deeper node, the sets A and B are empty for T_1 . Since all clashed facts are generated deterministically, the dependencies of the tuples have the current maximum branching tag 0 and are all collected into the set C . The backtracing continues with one tuple t from C , say $t = \langle B, k^{11,0}, x_1 \rangle$. The dependency k of t relates to the fact $(\forall r^-.B)(x_2)$, which is a part of the cause and replaces the backtraced tuple t in T_1 . The resulting set T_2 is used in the next loop round. The minimum node depth and the maximum branching tag remain unchanged, but the new tuple has a deeper node depth and is traced back with a higher priority to enable unsatisfiability caching again. Thus, $\langle (\forall r^-.B), i^{9,0}, x_2 \rangle$ is added to the set A and then replaced by its cause, leading to T_3 . Additionally, a pending creation of an unsatisfiability cache entry is noted, which is attempted in the third loop round since A and B are empty. The creation of a cache entry is, however, not yet sensible and deferred since T_3 still contains an atomic clash. Let $t = \langle B, j^{10,0}, x_1 \rangle \in C$ be the

$$\begin{aligned}
T_1 &= \{\langle \neg B, d^{4,0}, x_1 \rangle, \langle \neg B, e^{5,0}, x_1 \rangle, \langle B, j^{10,0}, x_1 \rangle, \langle B, k^{11,0}, x_1 \rangle\} \\
&\downarrow \\
T_2 &= \{\langle \neg B, d^{4,0}, x_1 \rangle, \langle \neg B, e^{5,0}, x_1 \rangle, \langle B, j^{10,0}, x_1 \rangle, \langle (\forall r^- . B), i^{9,0}, x_2 \rangle\} \\
&\downarrow \\
T_3 &= \{\langle \neg B, d^{4,0}, x_1 \rangle, \langle \neg B, e^{5,0}, x_1 \rangle, \langle B, j^{10,0}, x_1 \rangle, \langle (\exists r. (\forall r^- . B)), g^{7,0}, x_1 \rangle\} \\
&\downarrow \\
T_4 &= \{\langle \neg B, d^{4,0}, x_1 \rangle, \langle \neg B, e^{5,0}, x_1 \rangle, \langle r(x_1, x_2), h^{8,0}, x_1 \rangle, \langle (\exists r. (\forall r^- . B)), g^{7,0}, x_1 \rangle\} \\
&\downarrow \\
T_5 &= \{\langle \neg B, d^{4,0}, x_1 \rangle, \langle \neg B, e^{5,0}, x_1 \rangle, \langle (\exists r. (\forall r^- . B)), g^{7,0}, x_1 \rangle\} \\
&\downarrow \\
T_6 &= \{\langle \neg B, d^{4,0}, x_1 \rangle, \langle (\forall r. \neg B), -, a_0 \rangle, \langle (\exists r. (\forall r^- . B)), g^{7,0}, x_1 \rangle\} \\
&\downarrow \\
T_7 &= \{\langle r(a_0, x_1), b^{2,0}, x_1 \rangle, \langle (\forall r. \neg B), -, a_0 \rangle, \langle (A \sqcap (\exists r. (\forall r^- . B))), c^{3,0}, x_1 \rangle\} \\
&\downarrow \\
T_8 &= \{\langle (\exists r. (A \sqcap (\exists r. (\forall r^- . B))))-, a_0 \rangle, \langle (\forall r. \neg B), -, a_0 \rangle\}
\end{aligned}$$

Fig. 2. Backtracing sequence of tuples as triggered by the clash of Figure 1

tuple from T_3 that is traced back next. In the fourth round, the creation of a cache entry is attempted again, but fails because not all facts are concepts facts. The backtracing of $\langle r(x_1, x_2), h^{8,0}, x_1 \rangle$ then leads to T_5 . In the following round an unsatisfiability cache entry is successfully created for the set $\{\neg B, (\exists r. (\forall r^- . B))\}$. Assuming that now the tuple $\langle \neg B, e^{5,0}, x_1 \rangle$ is traced back, we obtain T_6 , which includes the node a_0 . Thus, the minimum node depth changes from 1 to 0. Two more rounds are required until T_8 is reached. Since all remaining facts in T_8 are concept assertions, no further backtracing is possible and an additional cache entry is generated for the set $\{(\exists r. (A \sqcap (\exists r. (\forall r^- . B))))-, (\forall r. \neg B)\}$.

If a tuple with a dependency to node a_0 had been traced back first, it would have been possible that the first unsatisfiability cache entry for the set $\{\neg B, (\exists r. (\forall r^- . B))\}$ was not generated. In general, it is not guaranteed that an unsatisfiability cache entry is generated for the node where the clash is discovered if there is no non-deterministic rule application and if the node is not a root node or an Abox individual. Furthermore, if there are facts that are not concept facts, these can be backtraced with higher priority, analogous to the elements of the set A , to make unsatisfiability cache entries possible again. To reduce the repeated backtracing of identical tuples in different rounds, an additional set can be used to store processed tuples for the alternative for which the backtracing is performed.

The backtracing can also be performed over nominal and Abox individual nodes. However, since Abox and absorbed nominal assertions such as $\{a\} \sqsubseteq C$ have no previous dependencies, this can lead to a distributed backtracing stuck on different nodes. In this case, no unsatisfiability cache entries are possible.

A less precise caching can lead to an adverse interaction with dependency directed backtracing. Consider the example of Figure 3, where the satisfiability of the combination of the concepts $(\exists r. (\exists s. (A \sqcap B)))$, $((C_1 \sqcap \forall r. C) \sqcup (C_2 \sqcap \forall r. C))$, and $((D_1 \sqcap \forall r. (\forall s. \neg A)) \sqcup (D_2 \sqcap \forall r. (\forall s. \neg A)))$ is tested. Note that, in order to keep the figure readable, we no longer show complete dependencies, but only the branching points for non-deterministic decisions. First, the two disjunctions are processed. Assuming that the alternative with the disjuncts $(C_1 \sqcap \forall r. C)$ and $(D_1 \sqcap \forall r. (\forall s. \neg A))$ is considered first (shown on the left-hand side of Figure 3), an r -successor x_1 with label $\{(\exists s. (A \sqcap B)), C^1, (\forall s. \neg A)^2\}$ is generated. The branching points indicate which concepts depend on which non-deterministic

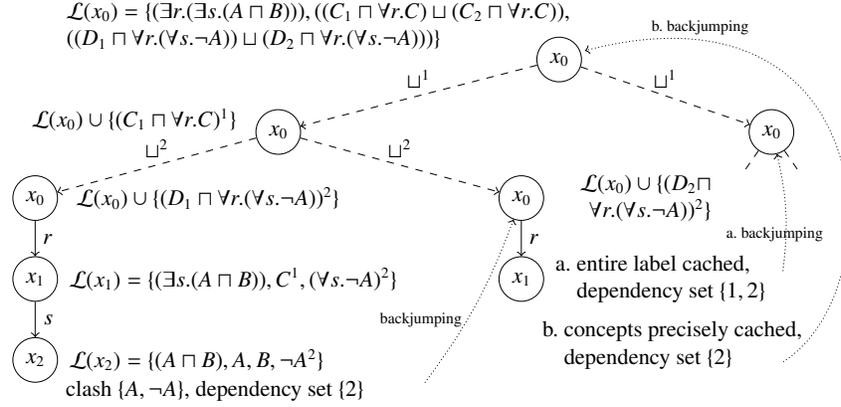


Fig. 3. More pruned alternatives due to dependency directed backtracking and precise caching (case b.) in contrast to label caching (case a.)

decision. For example, C is in $\mathcal{L}(x_1)$ due to the disjunct $(C_1 \sqcap \forall r.C)$ of the first non-deterministic branching decision (illustrated in Figure 3 with the superscript 1). In the further generated s -successor x_2 a clash is discovered. For the only involved non-deterministic branching point 2, we have to compute the second alternative. Thus, an identical r -successor x_1 is generated again for which we can discover the unsatisfiability with a cache retrieval. If the entire label of x_1 was inserted to the cache, the dependent branching points of all concepts in the newly generated node x_1 would have to be considered for further dependency directed backtracking. Thus, the second alternative of the first branching decision also has to be evaluated (c.f. Figure 3, case a., label caching). In contrast, if the caching was more precise and only the combination of the concepts $(\exists s.(A \sqcap B))$ and $(\forall s.\neg A)$ was inserted into the unsatisfiability cache, the cache retrieval for the label of node x_1 would return the inserted subset. Thus, only the dependencies associated to the concepts of the subset could be used for further backjumping, whereby it would not be necessary to evaluate the remaining alternatives (c.f. Figure 3, case b., precise caching).

4 Evaluation

Our Konclude reasoning system implements the enhanced optimisation techniques for *SROIQ* described above. We evaluate dependency directed backtracking and unsatisfiability caching with the help of concept satisfiability tests from the well-known DL 98 benchmark suite [10] and spot tests from [12]. An extended evaluation and a comparison of Konclude with other reasoners can be found in the accompanying technical report [15]. From the DL 98 suite we selected satisfiable and unsatisfiable test cases (with $_n$ resp. $_p$ postfixes) and omitted those for which unsatisfiability caching is irrelevant and tests that were too easy to serve as meaningful and reproducible sample.

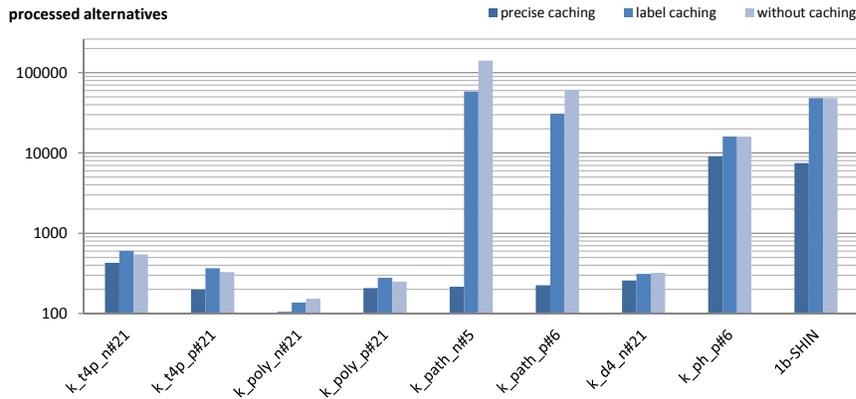


Fig. 4. Log scale comparison of processed alternatives for different caching methods

We distinguish between precise caching and label caching as described in Section 3.2. To recall, precise caching stores precise cache entries consisting of only those backtraced sets of concepts that are explicitly known to cause an unsatisfiability in combination with subset retrieval, while label caching stores and returns only entire node labels.

Konclude implements precise unsatisfiability caching based on hash data structures [8] in order to efficiently facilitate subset cache retrieval. Figure 4 shows the total number of processed non-deterministic alternatives for precise caching, label caching and without caching for a selection of test cases solvable within one minute.

Note that runtime is not a reasonable basis of comparison since the label caching has been implemented (just for the purpose of evaluation) on top of the built-in and computationally more costly precise caching approach. System profiling information, however, strongly indicate that building and querying the precise unsatisfiability cache within Konclude is negligible in terms of execution time compared to the saved processing time for disregarded alternatives. However, we have experienced an increase of memory usage by a worst-case factor of two in case of dependency tracking in comparison to no dependency handling.

Figure 4 reveals that precise caching can, for some test cases, reduce the number of non-deterministic alternatives by two orders of magnitude in comparison to label caching. Particularly the test cases k_path_n/p are practically solvable for Konclude only with precise caching for larger available problem sizes.

5 Conclusions

We have presented an unsatisfiability caching technique that can be used in conjunction with the very expressive DL *SROIQ*. The presented dependency management allows for more informed backjumping, while also supporting the creation of precise cache unsatisfiability entries. In particular the precise caching approach can reduce the num-

ber of tested non-deterministic branches by up to two orders of magnitude compared to standard caching techniques. The optimisations are well-suited for the integration into existing tableau implementations for *SROIQ* and play well with other commonly implemented optimisation techniques.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
2. Ding, Y., Haarslev, V.: Tableau caching for description logics with inverse and transitive roles. In: Proc. 2006 Int. Workshop on Description Logics. pp. 143–149 (2006)
3. Ding, Y., Haarslev, V.: A procedure for description logic *ALCFI*. In: Proc. 16th European Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'07) (2007)
4. Donini, F.M., Massacci, F.: EXPTIME tableaux for *ALC*. J. of Artificial Intelligence 124(1), 87–138 (2000)
5. Goré, R., Widmann, F.: Sound global state caching for *ALC* with inverse roles. In: Proc. 18th European Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'09). LNCS, vol. 5607, pp. 205–219. Springer (2009)
6. Haarslev, V., Möller, R.: Consistency testing: The RACE experience. In: Proceedings, Automated Reasoning with Analytic. pp. 57–61. Springer-Verlag (2000)
7. Haarslev, V., Möller, R.: High performance reasoning with very large knowledge bases: A practical case study. In: Proc. 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01). pp. 161–168. Morgan Kaufmann (2001)
8. Hoffmann, J., Koehler, J.: A new method to index and query sets. In: Proc. 16th Int. Conf. on Artificial Intelligence (IJCAI'99). pp. 462–467. Morgan Kaufmann (1999)
9. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SROIQ*. In: Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
10. Horrocks, I., Patel-Schneider, P.F.: DL systems comparison. In: Proc. 1998 Int. Workshop on Description Logics (DL'98). vol. 11, pp. 55–57 (1998)
11. Horrocks, I., Patel-Schneider, P.F.: Optimizing description logic subsumption. J. of Logic and Computation 9(3), 267–293 (1999)
12. Liebig, T.: Reasoning with OWL – system support and insights –. Tech. Rep. TR-2006-04, Ulm University, Ulm, Germany (September 2006)
13. Liebig, T., Steigmiller, A., Noppens, O.: Scalability via parallelization of OWL reasoning. In: Proc. Workshop on New Forms of Reasoning for the Semantic Web: Scalable & Dynamic (NeFoRS'10) (2010)
14. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2), 51–53 (2007)
15. Steigmiller, A., Liebig, T., Glimm, B.: Extended caching, backjumping and merging for expressive description logics. Tech. Rep. TR-2012-01, Ulm University, Ulm, Germany (2012), available online at http://www.uni-ulm.de/fileadmin/website_uni_ulm/iui/Ulmer_Informatik_Berichte/2012/UIB-2012-01.pdf
16. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proc. 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06). LNCS, vol. 4130, pp. 292–297. Springer (2006)
17. Tsarkov, D., Horrocks, I., Patel-Schneider, P.F.: Optimizing terminological reasoning for expressive description logics. J. of Automated Reasoning 39, 277–316 (2007)

From \mathcal{EL} to Tractable Existential Rules with Complex Role Inclusions

Michaël Thomazo

University Montpellier 2

Abstract. Ontology-based data access consists in using ontologies while querying data. Due to the high complexity of this problem, considering lightweight description logics like \mathcal{EL} is especially relevant. Another strand of research is based on existential rules. In this paper, we use this latter formalism in order to cover \mathcal{EL} with the same complexity of reasoning while allowing any predicate arity and some cycles on variables. We then add complex role inclusions to enhance expressivity, while staying polynomial in data complexity and generalizing existing results. In particular, we consider transitivity and right/left identity rules, which do not behave well with respect to usual decidability paradigms.

1 Introduction

Ontology-based data access (OBDA) recently received a lot of attention both from knowledge representation and database communities. This problem can be stated as follows: given a set of facts and an ontology (the knowledge base), one wants to evaluate a conjunctive query against this knowledge base. The ontology can be represented in several ways. Traditional ontology languages are description logics (DLs,[4]). The original focus of DL research was the ontology in itself, with problems like satisfiability or subsumption between concepts. The conjunctive query answering problem has been considered more recently. Classical DLs appeared to be highly complex for that reasoning problem, and lightweight description logics (such as \mathcal{EL} and DL-Lite) are thus very relevant.

A parallel approach relies on existential rules [5, 6], also known as TGDs [1] that form the basis of Datalog[±] [8]. Existential rules are logical formulas of the form $B \rightarrow H$, where B and H are conjunctions of atoms and where H might contain existentially quantified variables. In contrast to DLs, they allow for any predicate arity (which in particular eases the integration with database systems in which relations are naturally translated into n -ary predicates) and can express some cyclic dependencies on variables. On the other hand, neither disjunction nor negation is expressible with existential rules. Interestingly, the two main families of DLs that have been designed for conjunctive query answering can be translated into existential rules. The associated ontology-based conjunctive query answering problem (CQA) is formalized as follows: given a set of facts (in their logical form an existentially closed conjunction of atoms) F , a set of rules \mathcal{R} , and a conjunctive query Q , check whether $F, \mathcal{R} \models Q$ hold. This problem is undecidable, and numerous restrictions on \mathcal{R} have been proposed recently in order to ensure decidability (see [12] for a survey), which is usually proven by means of one

of the two following mechanisms, or a combination of both. The first one is forward chaining: rules are iteratively applied to F , until either no new information is added, or the query is entailed. If a set of rules is such that for any fact, this process halts in finite time or generates a set of facts of bounded treewidth (which is defined on a graph naturally associated with the facts, see e.g. [6]), then the CQA problem is decidable ([7, 5]). The second mechanism is backward chaining: a query Q is rewritten into a set of conjunctive queries (which can be seen as a union of conjunctive queries), such that Q is entailed by F and \mathcal{R} if and only if one its rewritings is entailed by F . The CQA problem is decidable if the set of rewritings is finite for any query.

\mathcal{EL} [2, 11] is one of the description logics that have a reasonable complexity for CQA: NP-complete in combined complexity and PTIME-complete in data complexity. As pointed out before, any \mathcal{EL} TBox can be translated into existential rules. However, the smallest known Datalog[±] decidable class covering such rules is a class for which CQA complexity is much higher than the original one (2-EXPTIME-complete in combined complexity). Finally, it is known that one can add to \mathcal{EL} inclusions a special kind of complex role inclusions while keeping polynomial data complexity [10]. As far as we know, such results have no counter-part in the rule framework. Moreover, one of the most used complex role inclusion, namely transitivity, is out of the scope of known decidability criteria when combined with decidable classes of existential rules.

The contribution of this paper is two-fold:

- first, it presents a class of existential rules, namely orientable *fr1*, that covers \mathcal{EL} ontologies while keeping the same (data or combined) complexity for CQA (Theorem 1). The proposed class allows for predicates of arbitrary arity and a form of cyclic dependencies between variables;
- second, it generalizes this class by adding complex role inclusions while staying polynomial in data complexity (Theorem 2); it allows for left and right identity rules, which have been proven useful for modeling purposes [3]. The syntactic regularity condition enforced is close from the one imposed in Horn-*SR1Q* ([9]).

In Section 2, basic definitions about \mathcal{EL} and existential rules are recalled. In Section 3, *orientable fr1 rules* are introduced. Section 4 adapts the algorithm presented in [13] and designed for a more general existential rule class, yielding an easier and worst-case optimal algorithm for orientable *fr1* rules. This algorithm is further modified in Section 5 in order to take some complex role inclusions into account. Section 6 concludes the paper.

In this paper, we will avoid technical definitions and rely on examples, to provide an intuition about the main techniques.

2 Preliminaries

We briefly recall the preliminary definitions presented in [6]. An atom is of the form $p(t_1, \dots, t_k)$ where p is a predicate with arity k , and the t_i are terms, i.e., variable or constants. A *fact* is the existential closure of a conjunction of atoms.¹ An *existential rule*

¹ Note that this notion generalizes the usual definition of fact by taking existential variables generated by rules into account.

(or simply a *rule* when not ambiguous) is a formula $R = \forall \mathbf{x} \forall \mathbf{y} (B[\mathbf{x}, \mathbf{y}] \rightarrow (\exists \mathbf{z} H[\mathbf{y}, \mathbf{z}]))^2$ where $B = \text{body}(R)$ and $H = \text{head}(R)$ are finite conjunctions of atoms, called the *body* and the *head* of R , respectively. The *frontier* of R , denoted by $\text{fr}(R)$, is the set of variables $\text{vars}(B) \cap \text{vars}(H) = \mathbf{y}$. A rule R is *applicable* to a fact F if there is a homomorphism π from $\text{body}(R)$ to F ; the result of the *application of R on F w.r.t. π* is a fact $\alpha(F, R, \pi) = F \cup \pi^{\text{safe}}(\text{head}(R))$ where π^{safe} is a substitution of $\text{head}(R)$, that replaces each $x \in \text{fr}(R)$ with $\pi(x)$, and each other variable with a “fresh” variable, i.e., not introduced before. The direct saturation of F with \mathcal{R} is defined as $\alpha(F, \mathcal{R}) = F \cup_{(R=(H,C),\pi) \in \Pi(F,\mathcal{R})} \pi^{\text{safe}}(C)$, where $\Pi(F, \mathcal{R}) = \{(R, \pi) \mid R = (B, H) \in \mathcal{R} \text{ and } \pi \text{ is a homomorphism from } H \text{ to } F\}$. The k saturation of F with \mathcal{R} is denoted by $\alpha_k(F, \mathcal{R})$ and is such that: $\alpha_0(F, \mathcal{R}) = F$, and for $i > 0$, $\alpha_i(F, \mathcal{R}) = \alpha(\alpha_{i-1}(F, \mathcal{R}), \mathcal{R})$. The universal model of F and \mathcal{R} is the union of $\alpha_k(F, \mathcal{R})$ for $k \in \mathbb{N}$. Q is entailed by F and \mathcal{R} iff it is entailed by $\alpha_k(F, \mathcal{R})$ for some $k \in \mathbb{N}$ (i.e., by the universal model of F and \mathcal{R}).

An \mathcal{EL} TBox contains concept inclusions $C_1 \sqsubseteq C_2$, where C_1 and C_2 are concepts built as follows:

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists R.C$$

Any \mathcal{EL} ontology can be translated into a set of existential rules (which are called \mathcal{EL} -rules), where $C_1 \sqsubseteq C_2$ is translated into $\phi(C_1)(x) \rightarrow \phi(C_2)(x)$, with ϕ inductively built as explained Table 1. The smallest known decidable class covering rules needed for the translation of an arbitrary \mathcal{EL} TBox is the set of frontier-1 (*fr1*) rules, i.e., the set of rules whose body and head share exactly one variable.

Table 1. Translation of an \mathcal{EL} ontology

\mathcal{EL} concept	Logical translation ϕ
A	$A(x)$
$C_1 \sqcap C_2$	$\phi(C_1)(x) \wedge \phi(C_2)(x)$
$\exists R.C$	$r(x, y) \wedge \phi(C)(y)$

3 Orientable *fr1* Rules

However, the combined complexity of reasoning with *fr1* rules and \mathcal{EL} is quite different: 2-EXPTIME-complete in the former case, and NP-complete in the latter – though both are in PTIME for data complexity. In this section, we present a class of rules that covers \mathcal{EL} while keeping the same combined and data complexities.

\mathcal{EL} -rules have the following idiosyncrasy: they add information “below” the frontier of the rule, as pictured in Figure 1. Any \mathcal{EL} -rule mapping its frontier to x_1 will map its body to dashed atoms, and will create atoms that are also below x_1 . This is due to the fact that information about a term “above” x_1 is not even expressible, since no inverse role is possible. *Orientable fr1 rules* generalize this idea.

We define for every predicate r of arity k a strict total order on its positions r^1, \dots, r^k . We denote by $<$ the union of all these relations. Given such a strict partial order, we can associate a directed graph with each set of atoms.

² We can now omit quantifiers since there is no ambiguity.

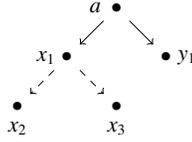


Fig. 1. An arc is directed from the first to the second argument of an atom.

Definition 1 (<-graph associated with a set of atoms). Let A be a set of atoms. The <-graph associated with A is the directed graph defined as follows:

- to each term x appearing in A , we assign a vertex v_x ,
- for any x, y such that $x \neq y$ and x and y appear in the same atom with position $p^x < p^y$, there is an arc from v_x to v_y (note that no loop can occur).

Intuitively, a rule is oriented for an order $<$ if the atoms needed to trigger it as well as the atoms created by it are situated both in the same right direction according to $<$.

Definition 2 (<-oriented fr1 rule). Let R be a rule and $<$ a strict total order on the position of its predicates. R is <-oriented fr1 if it is fr1 and if the <-graph associated with $\text{body}(R) \cup \text{head}(R)$ is a directed acyclic graph such that there is a path from $\text{fr}(R)$ to any other node.

Given this notion of rule orientation, we naturally define the notion of orientable fr1 set of rules.

Definition 3 (Orientable fr1 set of rules). A set of rules \mathcal{R} is orientable fr1 (or simply orientable when not ambiguous) if there exists an order $<$ such that every rule of \mathcal{R} is <-oriented fr1.

Example 1 (Orientable fr1 set of rules). Let us take $\mathcal{R} = \{r(x, y) \wedge p(y) \rightarrow q(x, z, t) \wedge s(z, t); q(x, y, z) \wedge s(y, z) \rightarrow r(x, t) \wedge r(t, t) \wedge p(t)\}$. By taking $r^1 < r^2, s^1 < s^2$ and $q^1 < q^2 < q^3$, we can easily check that this set of rules is <-oriented. Note that these rules are not translatable in \mathcal{EL} because of the predicate of arity 3 and of cycles.

Property 1 (Recognizability problem) The problem of deciding whether a set of rules \mathcal{R} is orientable is NP-complete.

The hardness result comes from a reduction of 3-SAT. NP-hardness of the recognizability problem might impede the practical applicability of following results. However, this complexity remains quite small compared to the combined complexity of CQA with a lot of known classes, and, more importantly, even strongly restricted sets of orientable rules are still of interest. Indeed, the next property shows that rules translating \mathcal{EL} ontologies are very naturally oriented (with $R^1 < R^2$ for any role R).

Property 2 (Generalization of \mathcal{EL}) Any set of \mathcal{EL} -rules is orientable.

In the next examples, we assume for all predicate p that $p^i < p^j$ if and only if $i < j$.

4 An Algorithm for CQA with Orientable Rules

In this section, we present a forward chaining algorithm which is a simplified version of the algorithm introduced in [13] for a class of rules called greedy bounded treewidth set, which includes *frl*. While performing forward chaining, a *greedy tree decomposition* (of bounded width) of the currently generated fact is maintained. We call *bags* the nodes of this tree, which is built as follows: the root of this tree contains all atoms in F , and each time a rule with frontier f is applied by means of a homomorphism π , we create a new bag that contains the newly generated atoms, and choose as its pBarent the root of the tree if $\pi(f)$ is a constant or a variable of F , otherwise the bag in which the variable $\pi(f)$ has been generated.

Example 2 (Greedy tree decomposition). Let $F = \{p(a), q(a)\}$ and $\mathcal{R} = \{R_1 : p(x) \rightarrow r(x, y) \wedge q(y); R_2 : q(x) \rightarrow t(x, y) \wedge p(y)\}$. We show the greedy tree decomposition of $\alpha_4(F, \mathcal{R})$ in Figure 2.

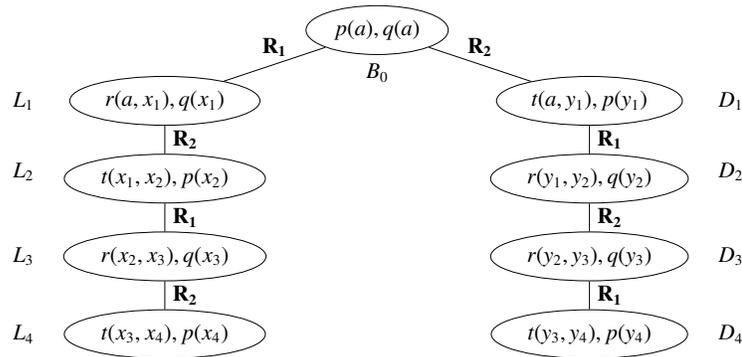


Fig. 2. The greedy tree decomposition of $\alpha_4(F, \mathcal{R})$ (Example 2)

Maintaining this tree decomposition is not sufficient by itself to ensure the termination of the algorithm, since the universal model can be infinite. The main notion used in [13] to ensure the finiteness of the built tree is the equivalence between bags: two bags B and B' are said equivalent when every fact that will eventually be mapped “under B ” (i.e., using at least one term generated below B) can be mapped similarly (i.e., up to a bijection between terms of B and B') “under B' ”, and conversely. If two bags are equivalent, it is only necessary to apply rules below one of them, and the other one will be said “blocked”. When no more rule is applicable on a non-blocked bag, we obtain the *full blocked tree*.

This equivalence as well as rule applications are computed in the original algorithm by means of *patterns*, that are attached to each bag. The complexity of the original algorithm is due to the high number of relevant patterns. In the remaining of this section,

we explain how to compute the equivalence relation as well as new rule applications without using patterns – and thus we do not present patterns here.

First, we can simplify equivalence between bags: with orientable *fr1* rules, two bags are equivalent if they have been created by the same rule, as stated by the following property.

Property 3 *Let \mathcal{R} be a set of orientable fr1 rules, $R \in \mathcal{R}, R' \in \mathcal{R}$, and F be a fact. Let B_1 and B_2 be two bags of \mathcal{T}^* (the tree decomposition of the universal model of F and \mathcal{R}) created by the same rule R . Let z be an existential variable of R , z_1 and z_2 be the corresponding fresh variables in B_1 and B_2 . B_1 has a child created by a rule application of R' mapping $\text{fr}(R')$ to z_1 if and only if B_2 has a child created by a rule application of R' mapping $\text{fr}(R')$ to z_2 .*

Example 2 (contd.). A full blocked tree of F and \mathcal{R} is represented in Figure 3. L_2 is equivalent to D_1 , D_2 to L_1 ; L_2 and D_2 are blocked and no new rule application can be done on B_0, L_1 or D_1 – this is checked by existence of a $*$ -homomorphism, see below.

Second, we check rule applicability by means of $*$ -homomorphism. This tool is introduced in [13] to evaluate a conjunctive query. Suppose that, at some step of the algorithm, we have generated a blocked tree \mathcal{T} . We want to check if there is a homomorphism from Q to the possibly infinite fact encoded by \mathcal{T} . This fact can be obtained from \mathcal{T} by a possibly infinite sequence of *completions*, that iteratively copies under blocked bags the atoms found under their equivalent bag (up to variable renaming). Such a homomorphism induces a partition of Q (two atoms are in the same set if they are mapped to the same – initial or added – bag) and a tree structure for this partition (mimicking the tree structure of the image bags). Finally, [13] shows that if there exists such a homomorphism, there exists one requiring only a completion sequence of bounded length. To encode the homomorphism, we thus only need the tree decomposition of Q , homomorphisms from each subset of Q to a bag of \mathcal{T} , and the required bounded number of completions: this is the structure called a $*$ -homomorphism.

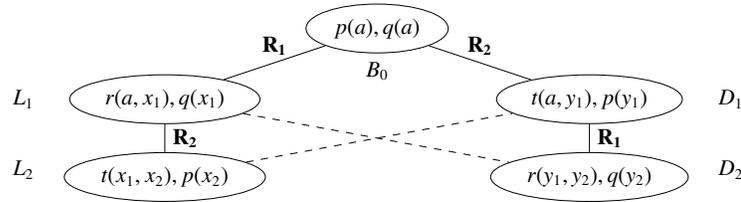


Fig. 3. The full blocked tree of F and \mathcal{R} (Example 2)

Example 3 ($$ -homomorphism).* Let $Q = \{r(z, z_1), t(z_1, z_2), r(z_2, z_3), t(z_3, z_4), t(z, z'_1)\}$. Let $\pi = \{(z, a), (z_1, x_1), (z_2, x_2), (z_3, x_3)(z_4, x_4), (z'_1, y_1)\}$ from Q to the fact associated with the tree drawn in Figure 2. The corresponding $*$ -homomorphism is pictured in Figure 4.

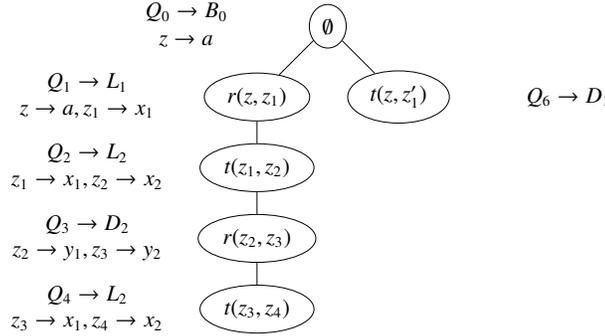


Fig. 4. A $*$ -homomorphism from $Q = \{r(z, z_1), t(z_1, z_2), r(z_2, z_3), t(z_3, z_4), t(z, z'_1)\}$ to the structure of Figure 3.

Given a tree decomposition of Q and homomorphisms from bags of this decomposition to corresponding bags of \mathcal{T} (as in Figure 4), we check if it is actually a $*$ -homomorphism. We check for any pair of adjacent nodes that the associated mappings are compatible, i.e., the image of any term is unique. In the given example, for Q_1 , $\{(z, a), (z_1, x_1)\}$ is a homomorphism from $r(z, z_1)$ to $r(a, x_1), q(x_1)$, which is consistent with Q_0 since z is mapped to a in both cases. An interesting consistency check occurs for Q_2 and Q_3 : since Q_2 has been mapped to a bag having no child, z_2 has image x_2 when considering Q_2 , and y_1 when considering Q_3 . However, this is consistent since when copying D_2 under L_2 , we rename y_1 by x_2 .

Theorem 1. *The conjunctive query answering problem with a set of orientable $fr1$ rules is PTIME-complete for data complexity and NP-complete for combined complexity.*

Proof. The PTIME membership comes from similar result for $fr1$ rules in [13]. The NP-membership is due to the fact that the structure we build is of polynomial size in F and \mathcal{R} . The hardness holds because it generalizes \mathcal{EL} .

It is interesting to note that our algorithm, when further restricted to \mathcal{EL} ontologies, becomes similar to the algorithm presented in [11]. The main difference is that, while our algorithm maintains equivalence classes, the algorithm in [11] *merges* equivalent bags. These merges result in the finiteness of the process, but also in the loss of homomorphism soundness with respect to first-order semantics. Soundness is regained by modifying homomorphism check, in a way both based on the orientation of \mathcal{EL} and on its tree-model property.

5 Adding Complex Role Inclusions

Complex role inclusions are arguably useful in practice. However, this kind of rules does not behave well with respect to both forward chaining and backward chaining, as illustrated with Example 4.

Example 4. Let $\mathcal{R} = \{p(x) \rightarrow r(x, y) \wedge p(y); r(x, y) \wedge r(y, z) \rightarrow r(x, z)\}$. The query $Q = \{q(x), r(x, y), q(y)\}$ is not rewritable w.r.t. \mathcal{R} into a finite union of conjunctive queries, and \mathcal{R} does not generate facts of bounded treewidth either: a clique of arbitrary size can be built by performing forward chaining on the fact $p(a)$.

In order to stay as close as possible to the algorithm of previous section, we split any fact generated during the chase into two parts: atoms generated by *fr1* rules (the *backbone*), and those generated by role inclusions. The first part is of bounded treewidth, and can be managed in the same way as before. The second part will be dealt with in a different fashion: we restrict the set of allowed role inclusions in order to deal with them by means of automata, as it was already done in [10]. In the following, we define an abstract condition of regularity for so-called oriented join rules, and an easily checkable syntactic condition that generalizes existing results. We then illustrate the algorithm with Example 5.

Definition 4 (Join rule). A join rule is a rule of the following shape: $r(x, y) \wedge s(y, z) \rightarrow t(x, z)$. Transitivity rules are the case where $r = s = t$, left-identity rules where $r = t$.

Definition 5 (Backbone). Let F be a fact, $\mathcal{R} = \mathcal{R}_o \cup \mathcal{R}_j$ where \mathcal{R}_o is a set of *fr1* rules and \mathcal{R}_j is a set of join rules. The backbone associated with F and \mathcal{R} is the subset of atoms of the universal model of F and \mathcal{R} that have been created by a *fr1* rule.

The backbone is of bounded treewidth, and a tree decomposition can be built greedily. In order to have a counter-part of Property 3 on bag equivalence, we consider only orientable join rules.

Definition 6 (<-oriented join rules). Let R be the following join rule: $r(x, y) \wedge s(y, z) \rightarrow t(x, z)$. R is a <-oriented join rule if:

- either $r^1 < r^2, s^1 < s^2$ and $t^1 < t^2$,
- or $r^1 > r^2, s^1 > s^2$ and $t^1 > t^2$.

The following property allows us to keep a trivial equivalence relation on bags. Indeed, it ensures that the criterion used in the previous section to check equivalence between bags remains valid when adding <-oriented join rules to our framework.

Property 4 Let \mathcal{R}_o be a set of <-oriented *fr1* rules, \mathcal{R}_j be a set of <-oriented join rules (for the same order <), $R, R' \in \mathcal{R}_o$, F be a fact. Let B_1 and B_2 two bags of \mathcal{T}^* (the greedy tree decomposition of the backbone associated with F and $\mathcal{R}_o \cup \mathcal{R}_j$) created by the same rule $R \in \mathcal{R}_o$. Let z be an existential variable of R , z_1 and z_2 be the corresponding fresh variables of B_1 and B_2 . B_1 has a child created by a rule application of R' mapping $\text{fr}(R')$ to z_1 if and only if B_2 has a child created by a rule application of R' mapping $\text{fr}(R')$ to z_2 .

Having a finite representation of the backbone, we use *regularity* in order to manage join rules. Given a fact F and x and y two terms of F , we denote by $\mathcal{P}_F(x, y)$ the set of words that describe a finite elementary path from x to y . For instance, in the query Q of Figure 4, $\mathcal{P}_F(z, z_3) = \{rtr\}$.

Definition 7 (Regularity of a set of join rules). Let P be a set of binary predicates, and \mathcal{R}_c be a set of join rules over P . \mathcal{R}_c is regular if for any predicate $p \in P$, there exists a regular language \mathcal{L}_p such that, for any fact F , for any $x, y \in \text{terms}(F)$, the following holds:

$$F, \mathcal{R}_c \models p(x, y) \Leftrightarrow \mathcal{P}_F(x, y) \cap \mathcal{L}_p \neq \emptyset$$

Similarly to a $*$ -homomorphism, a $*_j$ -homomorphism is a labeled tree structure, together with a mapping from its bags to the bags of the full blocked tree. In the $*$ -homomorphism case, the consistency check consists only in checking that each term of the query has a well-defined image. In the $*_j$ -homomorphism case, we also have to check that atoms generated by complex role inclusions can be effectively generated. Let us consider Example 5 and Figure 5. In order to have $w(a, x)$ entailed by the universal model, not only should x be mapped to t_1 in a bag B equivalent to B_3 , but there should also be a path from a to the image of the frontier of the rule creating B , such that \mathcal{A}_w (Figure 5) ends in s_3 when reading the word associated with that path. It is worth to note that this is not the case for B_3 and B_5 , but it holds for B_7 , even though B_3, B_5 and B_7 are equivalent. We thus add this information about states in the $*_j$ -homomorphism.

Compared to the orientable *frl* case, the size of a completion needed to map a query can be bigger up to a factor which is exponential in the query and in the automaton used to recognize regular predicates. This does not increase the data complexity of CQA with the union of a \leftarrow -oriented *frl* set of rules and a \leftarrow -oriented and regular set of join rules, which remains PTIME-complete. However further work is required to determine the combined complexity of the problem.

Example 5. Let $\mathcal{R} = \mathcal{R}_o \cup \mathcal{R}_j$ with $\mathcal{R}_o = \{p(x) \rightarrow r(x, y) \wedge r(y, z) \wedge q(z) \wedge p(z), q(x) \rightarrow w(x, y)\}$ and $\mathcal{R}_j = \{r(x, y) \wedge r(y, z) \rightarrow s(x, z), s(x, y) \wedge r(y, z) \rightarrow t(x, z), t(x, y) \wedge w(y, z) \rightarrow w(x, z)\}$. We take a fact $F = \{p(a)\}$ and a query $Q = \{w(a, x)\}$. The regular expressions associated with r, s, t and w are $r, rr + s, rrr + sr + t$, and $(rrr + sr + t)^*w$.

There exists a homomorphism π from Q to the completion represented in Figure 6, mapping x to t_3 . We represent this homomorphism by the structure represented Figure 7.

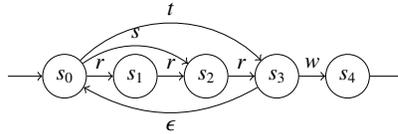


Fig. 5. \mathcal{A}_w , an automaton recognizing \mathcal{L}_w (Example 5)

Last, we present a syntactic condition that ensures that a set of join rules is regular. In particular, this condition generalizes the condition proposed in [10].

Definition 8 (Stratified set of join rules). A set of join rules is stratified if the directed graph G built as follows is acyclic. For every predicate p , there exists a vertex v_p in V . There is an arc from v_p to v_q where $p \neq q$ if there exists a rule $r(x, y) \wedge s(y, z) \rightarrow t(x, z)$, where $p = r$ or $p = s$ and $q = t$.

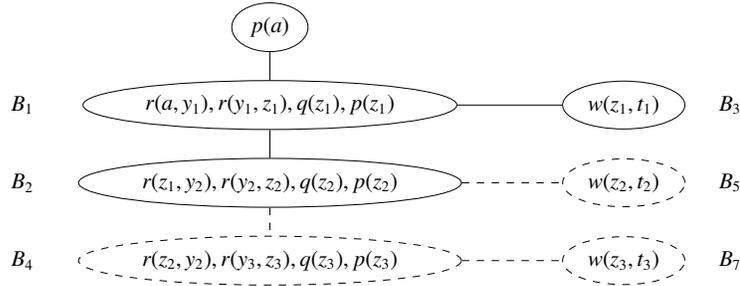


Fig. 6. In plain, the full blocked tree; in dashed, a completion (Example 5). By additionally using (only) role inclusions, $Q = w(a, x)$ can be mapped, mapping x to t_3 .

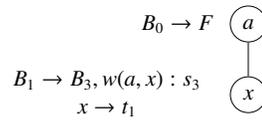


Fig. 7. A $*_j$ -homomorphism of $Q = \{w(a, x)\}$ in the full blocked tree of Example 5

Property 5 (Regularity of a stratified set of join rules) *A stratified set of join rules is regular.*

Proof (sketch). By induction on the structure of G . A predicate p whose vertex does not have an incoming arc is associated with the regular expression p . For any q , if we have a regular expression for any p such that (v_p, v_q) is an arc of G , we can build one for q .

Theorem 2. *The CQA problem with rules being the union of of \leftarrow -oriented fr1 rules and \leftarrow -oriented regular join rules (for the same order \leftarrow) is PTIME in data complexity.*

6 Conclusion

In this preliminary work, we identified a class of existential rules, namely *fr1* orientable rules, covering \mathcal{EL} ontologies while keeping the same complexity for CQA. Rules allow to easily use predicates of any arity and some cycles between variables. We exploited the simplicity of orientable *fr1* rules to simplify the very recent algorithm from [13]. We then investigated how to add some expressive power that could be useful in practice, while staying polynomial in data complexity: we showed how to add transitivity axioms and left- and right-identity rules. Although adding these rules does not fulfill the usual decidability criteria, we adapted the algorithm for orientable rules by using finite automata. Some follow-up naturally come to mind: can we generalize our approach to cover Horn-*SROIQ*? What is the combined complexity of CQA with the considered rules? What are interesting classes of rules with predicate of any arity that could be added to this basis, while staying polynomial in data complexity? How does the algorithm behave in practice? Moreover, compared to the algorithm presented in [11], the

representation of the universal model uses more space, since equivalent nodes are not merged. Can we adapt our algorithm in order to reduce the space requirements?

Acknowledgment

The author thanks the anonymous reviewers for their useful and constructive comments.

References

1. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*. Addison Wesley (1994)
2. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: IJCAI. pp. 325–330 (2003)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 364–369. Professional Book Center (2005)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edn. (2007)
5. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. In: IJCAI (2009)
6. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. *Artif. Intell.* 175(9-10), 1620–1654 (2011)
7. Cali, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008), Sydney, Australia. pp. 70–80 (2008)
8. Cali, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Proceedings of the 28th symposium on Principles of database systems (PODS'09). pp. 77–86. ACM, New York, NY, USA (2009), <http://doi.acm.org/10.1145/1559795.1559809>
9. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: KR. pp. 57–67 (2006)
10. Krötzsch, M., Rudolph, S.: Conjunctive queries for \mathcal{EL} with role composition. In: DL. vol. 250 (2007)
11. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic \mathcal{EL} using a relational database system. In: IJCAI (2009)
12. Mugnier, M.L.: Ontological query answering with existential rules. In: RR. pp. 2–23 (2011)
13. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: KR ((to appear) 2012)

Logical Relevance in Ontologies

Chiara Del Vescovo, Bijan Parsia, and Uli Sattler

University of Manchester, UK

Abstract Most ontology development environments (ODEs) are term oriented and take a frame-based view of the information in an ontology about a given term. Even tools, such as Protégé 4, designed for axiom oriented development preserve the frame-based view as the central mode of interaction with the ontology. The frame-based approach has a number of advantages—most prominently that it is comfortable to people familiar with object oriented programming languages. However, in expressive languages the frame-based views suffer from being only sensitive to syntactic relations between axioms and terms, thus possibly missing key logical relations.

In this paper, we first introduce a semantic notion of relevance between a term and axioms in an ontology, and we investigate the relation of this concept with the inseparability relation based on model Conservative Extensions. Unfortunately, we cannot use model conservativity to detect relevance since it is hard, or even impossible, to decide. Hence, we approximate model conservativity using two notions of modules based on locality, that can be efficiently computed, and provide logical guarantees, e.g. they preserve entailments over a given signature. In particular, we define relevance via Atomic Decomposition, that is a dependency graph showing the logical relations enforced by the two notions of modules between the axioms. We define a suitable labelling that allows us to locate axioms that are relevant for a term in the AD dependency structure. Finally, we describe an interesting consequence of such a view in terms of the models of an ontology.

1 Introduction

Most ontology development environments (ODEs) are term oriented and take a frame-based view of the information in an ontology about a given term.¹ Even tools, such as Protégé 4, designed for axiom oriented syntaxes (such as the functional syntax of OWL 2) preserve the frame-based view as the central mode of interaction with the ontology. The frame-based approach has a number of advantages—most prominently that it is comfortable to people familiar with object oriented programming languages.

However, in expressive languages the frame-based view is prone to present a misleading view of what is relevant for a given term in an ontology: in particular, axioms which are relevant for the meaning of the term are excluded from such a view, and some extraneous ones may creep in. The main reason for this problem to occur is that frame-based views are generally only sensitive to syntactic relations between an axiom and a term, and thus they can miss key logical relations.

¹ We use *term* to mean any individual, concept, or role name belonging to the signature of the ontology.

Given an axiom α , it is easy to check whether it is logically relevant for a term \mathfrak{t} : if it constrains the meaning of \mathfrak{t} . As an example, let us consider the axiom $\alpha = 'A \sqsubseteq B \sqcap (C \sqcup \neg C)'$. Then, α is clearly relevant for both A and B, because it states a subsumption relation between the two concept names. However, α “does not say anything” about C, since for any interpretation $\mathcal{C}^{\mathcal{I}}$, the expression $C \sqcup \neg C$ is equivalent to \top , and can be discarded from the axiom obtaining an axiom $\alpha' = 'A \sqsubseteq B'$ logically equivalent to α . Hence, axioms irrelevant for \mathfrak{t} can easily sneak into the usage view. Tautologies as $\mathfrak{t} \sqsubseteq \top$, which are automatically generated when a new top class name is entered in an OWL ontology using Protégé 4, are a quite common example.

Another logical relation that we want to preserve concerns the consequences that *sets* of axioms can impose on a term. An issue in the detection of what contributes to the meaning of a term is the fact that a given term \mathfrak{t} does not even need to occur in an axiom’s signature for being constrained by it. As an example let us consider the set of axioms $\{\alpha_i = 'A_{i-1} \sqsubseteq A_i'\}_{i=1..n}$. Then, it is easy to see that, for $n \geq 2$ and $i = 2, \dots, n - 2$, both axioms α_1 and α_n are logically relevant for A_i even though their signatures do not contain it. Note that a complex logical interaction can occur also within ontologies with limited expressivity. However, logical relevance is clearly more interesting for more complex ontologies than taxonomies.

This paper is a preliminary investigation of the notion of relevance of axioms for a term under a model-theoretic perspective. The major aim of our future work consists of identifying an efficiently computable way to reveal the logical interactions between axioms and terms. The main applications of this study can be found in the areas for improving reasoners performance, and in supporting ontology engineers during the modelling process.

2 Preliminaries

We assume the reader to be familiar with Description Logics [1]. As usual in this context, we use \mathcal{O} for ontologies, i.e. finite sets of axioms based on a Description Logic, e.g., *SHIQ*, and $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for interpretations of \mathcal{O} over the domain $\Delta^{\mathcal{I}}$. We use $\tilde{\alpha}$ for the signature of an axiom α , i.e., the set of concept, role, and individual names used in α . A generic term \mathfrak{t} is any non logical atomic symbol of the signature $\tilde{\mathcal{O}}$ of the ontology. Given a signature $\Sigma \subseteq \tilde{\mathcal{O}}$, we denote by $\mathcal{I}|_{\Sigma}$ the restriction of the interpretation function \mathcal{I} over the symbols in Σ .

In this section we briefly summarize the key concepts used in the paper, as model conservativity [8], locality-based modules [3], Atomic Decompositions (ADs) [6] and their labelled versions (LADs) [4], plus some notions inherited by the algebraic order theory.

Model inseparability Two ontologies $\mathcal{O}_1, \mathcal{O}_2$ are *model-inseparable* w.r.t. a signature Σ —denoted $\mathcal{O}_1 \equiv_{\Sigma}^{mCE} \mathcal{O}_2$ —if $\{\mathcal{I}|_{\Sigma} \mid \mathcal{I} \models \mathcal{O}_1\} = \{\mathcal{J}|_{\Sigma} \mid \mathcal{J} \models \mathcal{O}_2\}$. We can then define an *mCE-module* w.r.t. a signature Σ to be a minimal set of axioms $\mathcal{M} \subseteq \mathcal{O}$ such that, for each model \mathcal{I} of \mathcal{M} , there is a model \mathcal{J} of \mathcal{O} such that $\mathcal{J}|_{\Sigma} = \mathcal{I}|_{\Sigma}$. Another notion we use is the *\mathfrak{t} -variant* of an interpretation \mathcal{I} , defined as an interpretation \mathcal{J} such that, for each symbol $s \in \tilde{\mathcal{O}} \setminus \mathfrak{t}$, we have $s^{\mathcal{I}} = s^{\mathcal{J}}$.

Locality-based modules Unfortunately, deciding if a set of axioms is an mCE module is hard or even impossible for expressive DLs [8,12]. Thus, efficiently computable approximations have been devised, as those defined via the notion of *syntactic locality*. A locality-based module \mathcal{M} for a signature Σ is an approximation of the mCE-module for Σ , in the sense that it is a (not minimal, but generally small) set of axioms that preserves all models over Σ . Interestingly, \mathcal{M} also preserves all entailments over Σ , even though possibly not only those. Locality-based modules are particularly interesting because the extraction of a module can be performed in polynomial time. We give an intuition of the definition in what follows, and refer the interested reader to [3] for a deeper discussion.

Intuitively, an axiom α is (syntactically) local w.r.t. a signature Σ if there is no axiom over Σ that is entailed by α . Locality is anti-monotonic, that is, if an axiom is non local w.r.t. Σ , then it is non local also w.r.t. any Σ' that contains Σ . So we can define a *minimal seed signature* for an axiom α to be a signature Σ such that α is non local w.r.t. Σ but it is local w.r.t. any proper subset of Σ .

Locality comes in two flavours: \perp and \top . Intuitively, an axiom is \perp -local w.r.t. a term when it fails to constrain it “from above”. As an example, let us consider $\alpha = \text{‘}A \sqsubseteq B\text{’}$; then, α is local w.r.t. B because, for any interpretation \mathcal{I} over $\{A\}$, \mathcal{I} can be extended by interpreting B as $\Delta^{\mathcal{I}}$ and still $\mathcal{I} \models \alpha$. Similarly, α is \top -local w.r.t. A because it fails to constrain A “from below”.

Locality-based modules then inherit a similar intuition: roughly speaking, a \perp -module for Σ (denoted $\perp\text{-mod}(\Sigma, \mathcal{O})$), when non empty, gives a view “from above” because it contains all subconcepts of concept names in Σ ; a \top -module for Σ (denoted $\top\text{-mod}(\Sigma, \mathcal{O})$) gives a view “from below” since it contains all superconcepts of concept names in Σ . Please note that \mathcal{M} is not simply the union of all non-local axioms w.r.t. Σ . The extraction algorithm is described in [3], and a module extractor based on syntactic locality is available in the OWL API.²

(Labelled) Atomic Decomposition The number of modules of an ontology \mathcal{O} can be exponential in the minimum amongst the number of axioms of \mathcal{O} and the size of its signature [14]. However we can focus on *genuine* modules, i.e. modules that are not the union of two “ \subseteq ”-uncomparable modules. Such modules define a base for all modules, and interestingly the size of the family of genuine modules for \mathcal{O} is linearly dependent on its size [6].

Some sets of axioms never split across two modules [6], revealing a strong logical interrelation. The notion of *Atomic Decomposition* provided next is central to our paper.

Definition 1. For $x \in \{\top, \perp\}$, we call x -atom a maximal set $\alpha^x \subseteq \mathcal{O}$ such that, for each x -module \mathcal{M}^x , either $\alpha^x \subseteq \mathcal{M}^x$, or $\alpha^x \cap \mathcal{M}^x = \emptyset$. The family of x -atoms of \mathcal{O} is denoted by $\mathcal{A}(\mathcal{O})$ and is called x -Atomic Decomposition (x -AD).

If the module notion x is clear from the context, or irrelevant, we drop it.

Since every atom is a set of axioms, and atoms are pairwise disjoint, the AD is a partition of the ontology, and its size is at most linear w.r.t. the size of the ontology. In particular, each axiom³ α belongs to one and only one atom, denoted α_α .

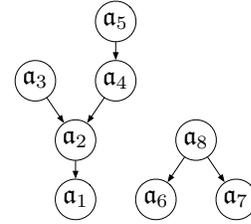
² <http://owlapi.sourceforge.net>

³ Syntactic tautologies do not occur in any atom; however, since they do not impose any constraint on the terms of an ontology \mathcal{O} because they are always true, we can safely remove them from \mathcal{O} and only consider the case where \mathcal{O} does not contain any such axioms.

Interestingly, there is a 1-1 correspondence between atoms and genuine modules: for each atom a we denote by \mathcal{M}_a the corresponding genuine module, that is also the smallest module containing a . Then we can define a second logical relation between atoms: an atom a is *dependent* on a distinct atom b (written $a \succ b$) if $\mathcal{M}_b \subseteq \mathcal{M}_a$. Note that this property then holds for all modules containing a . The dependence relation \succ on AD is a poset (i.e., transitive, reflexive, and antisymmetric), thus can be represented by means of a Hasse diagram. Moreover, it is computable in polynomial time [6]. To easy the understanding of what an AD of an ontology is, the Example 1 illustrates a small ontology and its \perp -AD.

Example 1. Consider the following toy ontology and its \perp -AD:

- (α_1) $\text{Animal} \sqsubseteq \exists \text{hasGender.Thing}$,
- (α_2) $\text{Animal} \sqsubseteq \geq \text{lhasHabitat.Thing}$,
- (α_3) $\text{Person} \sqsubseteq \text{Animal}$,
- (α_4) $\text{Vegan} \equiv \text{Person} \sqcap \forall \text{eats.}(\text{Vegetable} \sqcup \text{Mushroom})$,
- (α_5) $\text{Student} \sqsubseteq \text{Person} \sqcap \exists \text{hasHabitat.University}$,
- (α_6) $\text{GraduateStudent} \equiv \text{Student} \sqcap \exists \text{hasDegree.}(\{\text{BA}, \text{BS}\})$,
- (α_7) $\text{Car} \sqsubseteq \text{Vehicle}$,
- (α_8) $\text{Truck} \sqsubseteq \text{Vehicle}$,
- (α_9) $\text{Car} \sqsubseteq \neg \text{Truck}$



Here the \perp -atoms in the AD contain the following axioms respectively:

$\mathbf{a}_1 = \{\alpha_1, \alpha_2\}$, $\mathbf{a}_2 = \{\alpha_3\}$, $\mathbf{a}_3 = \{\alpha_4\}$, $\mathbf{a}_4 = \{\alpha_5\}$, $\mathbf{a}_5 = \{\alpha_6\}$, $\mathbf{a}_6 = \{\alpha_7\}$, $\mathbf{a}_7 = \{\alpha_8\}$, and $\mathbf{a}_8 = \{\alpha_9\}$.

Atoms can be seen as building blocks for modules: for each x -module \mathcal{M} of an ontology \mathcal{O} , there are atoms $\mathbf{a}_1, \dots, \mathbf{a}_\kappa$ in $\mathcal{A}(\mathcal{O})$ such that $\mathcal{M} = \bigcup_{i=1}^{\kappa} \mathbf{a}_i$. The converse does not hold, since not all combinations of atoms are modules. However, in [5] we studied and implemented an algorithm to extract modules of ontologies directly from their ADs, that is without loading the ontology. We use an enriched version of the ADs, called *Labelled Atomic Decomposition* (LAD), where each atom \mathbf{a} is mapped to the set minimal sets Σ of terms that make \mathbf{a} be included in the module for Σ . Depending on the task we want to use LADs for, different labels can be defined. A first investigation of tasks and suitable labels can be found in [4].

Order theory The poset structure induced over the atoms of an ontology allows us to take advantage of some useful algebraic notions. Given an atom \mathbf{a} , we define its *principal ideal* $\downarrow \mathbf{a}$ to be the set union of \mathbf{a} with all atoms \mathbf{b} such that $\mathbf{a} \succ \mathbf{b}$.⁴ Similarly, we can define the *principal filter* $\uparrow \mathbf{a}$ of \mathbf{a} as the set union of all atoms \mathbf{c} such that $\mathbf{c} \succ \mathbf{a}$. More in general, given a set \mathcal{S} of atoms $\{\mathbf{a}_1, \dots, \mathbf{a}_\ell\}$ we can define its ideal (filter) to be the union of the principal ideals (filters) of the atoms in \mathcal{S} . The usefulness of these algebraic notions for ADs is proven by the ease of getting the genuine module of an atom $\mathbf{a} \sqsubseteq \mathcal{O}$ from the AD of \mathcal{O} : we can just extract the principal ideal $\downarrow \mathbf{a}$.

⁴ Slightly abusing the notation, we define ideals as the union over poset elements rather than sets of poset elements. This choice allows ideals to be set of axioms, hence ontologies.

3 Semantic-based Relevance

While the semantic of the terminology of an ontology defines the objects that the ontology deals with, it does not say how these objects are related. The relationship between terms is defined by the *axioms* of the ontology, that constrain which interpretations are allowed, and which are not. For this reason, we are interested in looking for a semantic notion of relevance of an axiom for a term. In particular, the interpretations of two distinct terms can be conflicting only if some axioms are then violated. In this perspective, the natural choice is to investigate the notion of relevance of an axiom for a term, rather than relevance between terms. First, we introduce the following useful notions.

Definition 2. *Given a consistent ontology \mathcal{O} and a signature $\Sigma \subseteq \tilde{\mathcal{O}}$, we define a Σ -model w.r.t. \mathcal{O} to be an interpretation \mathcal{I} over Σ such that, there exists a model \mathcal{J} for \mathcal{O} such that $\mathcal{J}|_{\Sigma} = \mathcal{I}$. In this case, we say that \mathcal{I} is extendable to a model \mathcal{J} for \mathcal{O} , and any such \mathcal{J} is called an \mathcal{O} -extension of \mathcal{I} .*

If \mathcal{O} is clear from the context, we simply drop it and say Σ -model. Please also note that Def. 2 is also valid in the case of \mathcal{O} being a single axiom.

A first very basic notion of relevance is introduced in the following example: let α be the axiom $A \sqsubseteq B \sqcap (C \sqcup \neg C)$. Then, in order for an interpretation \mathcal{I} to be a model of α , it needs to satisfy the relation $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$. In other words, the acceptable interpretations of both A and of B are constrained. On the contrary, the interpretation of C is not constrained by α , and we can even rewrite the axiom into the equivalent $A \sqsubseteq B$ that does not even mention C. This case of relevance provided by a single axiom is described in the next definition.

Definition 3. *An axiom α is directly relevant to a term \mathfrak{t} if there exists a model \mathcal{I} of α and a $\{\mathfrak{t}\}$ -variant \mathcal{I}' of \mathcal{I} such that $\mathcal{I}' \not\models \alpha$.*

Note that if an axiom α does not contain a term \mathfrak{t} in its signature, then it does not directly constrain it. However, the inverse implication does not hold as we saw before.

An interesting strong relation between the notion of direct relevance and the notion of model conservativity is discussed in Prop. 1.

Proposition 1. *Let α be a consistent axiom and $\mathfrak{t} \in \tilde{\alpha}$ a term. If there exists a signature $\Sigma \subseteq \tilde{\alpha}$ such that $\Sigma \ni \mathfrak{t}$, $\alpha \not\equiv_{\Sigma}^{mCE} \emptyset$, and $\alpha \equiv_{\Sigma \setminus \{\mathfrak{t}\}}^{mCE} \emptyset$, then α is directly relevant for \mathfrak{t} .*

Proof. Let α be an axiom and \mathfrak{t} be a term such that there exists a signature $\Sigma \subseteq \tilde{\alpha}$ satisfying the hypothesis. Since $\alpha \not\equiv_{\Sigma}^{mCE} \emptyset$, we know that there exists an interpretation \mathcal{J} over Σ that cannot be extended to a model for α . In contrast, since $\alpha \equiv_{\Sigma \setminus \{\mathfrak{t}\}}^{mCE} \emptyset$, we have that $\mathcal{J}|_{\Sigma \setminus \{\mathfrak{t}\}}$ can be extended to a model \mathcal{I} for α . Let us now consider the interpretation \mathcal{J}' that interprets all symbols in $\Sigma \setminus \{\mathfrak{t}\}$ as \mathcal{I} does, whilst it interprets \mathfrak{t} as \mathcal{J} does. Then, \mathcal{J}' is not a model for α , and it is a \mathfrak{t} -variant for \mathcal{I} . Hence, α is directly relevant for \mathfrak{t} . \square

The inverse implication in Prop. 1 does not hold in general. For example, let us consider the axiom $\{a\} \sqsubseteq A$. Then, there is no model that interprets the symbol A as the empty set. In particular, $\alpha \not\equiv_{\emptyset}^{mCE} \emptyset$. However, the inverse implication can still hold

for less expressive DL than *SHROIQ*. An investigation on the characterization of the languages for which the inverse of Prop. 1 is part of our future work.

Let us now recall the trivial example in the introduction, where $\mathcal{O} = \{\alpha_i\}_{i=1\dots n}$ and the i -th axiom is $A_{i-1} \sqsubseteq A_i$. If $n \geq 2$ then the axiom α_n does not contain A_0 . However, α_n does *indirectly* constrain A_0 because in any model \mathcal{I} of \mathcal{O} where $A_{n-1}^{\mathcal{I}} = \emptyset$ (and α_n is then satisfied for any interpretation of A_n) we have that $A_0^{\mathcal{I}}$ is forced to be empty. In other words, in order to define irrelevance between a term and an axiom we need then to look at their interpretations *in the context of the ontology*. Intuitively, an axiom α is irrelevant for \mathfrak{t} w.r.t. the ontology \mathcal{O} if the interpretation of \mathfrak{t} and the interpretation of symbols in $\tilde{\alpha} \setminus \mathfrak{t}$ can be chosen independently from each other, even though we still have to take into account the constraints provided by \mathcal{O} . This idea is formalised in Def. 4.

Definition 4. *Let \mathcal{O} be an ontology, $\mathfrak{t} \in \tilde{\mathcal{O}}$ a term, and $\alpha \in \mathcal{O}$ an axiom. We say that α is \mathcal{O} -irrelevant for \mathfrak{t} if, for any $\{\mathfrak{t}\}$ -model $(\Delta^{\mathcal{I}_1}, \mathcal{I}_1)$ w.r.t. $\mathcal{O} \setminus \alpha$ and any $\{\tilde{\alpha} \setminus \mathfrak{t}\}$ -model $(\Delta^{\mathcal{I}_2}, \mathcal{I}_2)$ w.r.t. \mathcal{O} , there exists a model \mathcal{J} which is an \mathcal{O} -extension of both \mathcal{I}_1 and \mathcal{I}_2 . We say that α is \mathcal{O} -relevant for \mathfrak{t} if it is not \mathcal{O} -irrelevant.*

In many cases, if an axiom $\alpha \in \mathcal{O}$ is directly relevant for a term \mathfrak{t} , then α is also \mathcal{O} -relevant for \mathfrak{t} . However, this condition fails to hold when both α and $\mathcal{O} \setminus \alpha$ imply that there is only one valid $\{\mathfrak{t}\}$ -model w.r.t. \mathcal{O} . Such a peculiar case is described in the following example: let us consider the ontology $\mathcal{O} = \{\mathfrak{t} \sqsubseteq \perp, \mathfrak{t} \sqsubseteq A \sqcap \neg A\}$. Now, both axioms are clearly directly relevant for \mathfrak{t} . However, both of them are not \mathcal{O} -relevant for \mathfrak{t} , because we cannot find any $\{\mathfrak{t}\}$ -model w.r.t. \mathcal{O} where \mathfrak{t} is interpreted differently. Hence, we need the following unifying notion of relevance.

Definition 5. *An axiom $\alpha \in \mathcal{O}$ is said to be relevant for a term $\mathfrak{t} \in \tilde{\mathcal{O}}$ if α is either directly relevant or \mathcal{O} -relevant for \mathfrak{t} . Otherwise, α is said to be irrelevant for \mathfrak{t} .*

Note that this notion of relevance is still defined in the context of the ontology \mathcal{O} .

In the following, we denote:

1. the set of (semantically) directly relevant axioms w.r.t. a term \mathfrak{t} by $semDR_{\mathcal{O}}(\mathfrak{t}) = \{\alpha \in \mathcal{O} \mid \alpha \text{ is directly relevant for } \mathfrak{t}\}$
2. the set of (semantically) relevant axioms w.r.t. \mathfrak{t} by $semRel_{\mathcal{O}}(\mathfrak{t}) = \{\alpha \in \mathcal{O} \mid \alpha \text{ is relevant for } \mathfrak{t}\}$.

In this paper we are not interested in investigating the complexity for deciding relevance of an axiom to a term. Our aim is to use modules based on syntactic locality to efficiently get the two approximations $DC_{\mathcal{O}}(\mathfrak{t})$ for the set $semRel_{\mathcal{O}}(\mathfrak{t})$, and $C_{\mathcal{O}}(\mathfrak{t})$ for $semDR_{\mathcal{O}}(\mathfrak{t})$. By approximation we mean that *all* relevant axioms are preserved, even if some irrelevant axiom can sneak into such sets. This is still ongoing research, and from this point on the reader will find only definitions, examples, and conjectures. Proving these results is included in our future work.

4 Locality-based Relevance

In [7] we have analysed several forms of modularity to detect logically coherent subsets of an ontology (and more in general of a logical theory). The idea behind that paper

is that each kind of module determines a granular structure in the ontology, and this identifies clusters of axioms that stick together and axioms that can be separated from those clusters. Some of these notions of modularity relate also each cluster of axioms to a subset of the vocabulary used—hence relating axioms to terms. However, all these kinds of modularity suffer from inducing a coarse notion of internal coherence, and in some notable examples the ontology cannot be decomposed into smaller bits, even if it seems to be well structured.

In the same paper, we also analyse the partitioning of an ontology provided by one of its ADs. Atoms are generally very small bits, as discussed in [5], hence in principle such bits do not suffer from aggregating together (too many) unrelated axioms. However, ADs did not come with a semantically-based notion of relevance between the atoms and the terms of an ontology.

In what follows we define two labelling functions, the first mapping each axiom to relevant terms, and the second mapping each atom to the set of relevant terms. Then, we describe and conjecture the relations of the resulting LAD with the two sets $semRel_{\mathcal{O}}(\tau)$ and $semDR_{\mathcal{O}}(\tau)$, defined in the previous paragraph, that contain the relevant and the directly relevant axioms to a term.

Definition 6. *Let α be an axiom, and let τ be a term such that $\tau \in \tilde{\alpha}$. We say that:*

1. α constrains τ from above if there exists a minimal seed signature containing τ that makes α non \perp -local
2. α constrains τ from below if there exists a minimal seed signature containing τ that makes α non \top -local
3. α constrains τ if α constrains τ either from above, or from below.

We denote:

- by $C_{\mathcal{O}}^{\perp}(\tau)$ the set of axioms in \mathcal{O} constraining a term τ from above
- by $C_{\mathcal{O}}^{\top}(\tau)$ the set of axioms in \mathcal{O} constraining a term τ from below
- by $C_{\mathcal{O}}(\tau)$ the set of axioms in \mathcal{O} constraining a term τ .

We conjecture that the notion of constraining a term is an approximation of the notion of direct relevance.

Conjecture 1. Let τ be a term in the signature of the ontology \mathcal{O} . Then,

$$semDR_{\mathcal{O}}(\tau) \subseteq C_{\mathcal{O}}(\tau).$$

5 LADs and the Double Cones of Relevance

The next step is to relate the set $semRel_{\mathcal{O}}(\tau)$ to a suitable efficiently computable approximation. The idea is to identify in the ADs of the ontology the axioms constraining a term, and then follow along the ADs how the consequences logically “span” across the whole ontology. Specifically, we define the following labelling functions, and we conjecture that the resulting LADs are able to keep track of the logical relations between the axioms of an ontology \mathcal{O} and the terms in $\tilde{\mathcal{O}}$.

Definition 7. *Let α be an axiom in the ontology \mathcal{O} . We define the following labelling functions:*

1. $lab^\perp : \mathcal{O} \rightarrow \wp(\tilde{\mathcal{O}})$ that maps each axiom to the set of all terms that are constrained by α from above
2. $lab^\top : \mathcal{O} \rightarrow \wp(\tilde{\mathcal{O}})$ that maps each axiom to the set of all terms that are constrained by α from below
3. for any notion of module $x \in \{\perp, \top\}$, $Lab^x : \mathcal{A}^x(\mathcal{O}) \rightarrow \wp(\tilde{\mathcal{O}})$ that maps each atom a to the set $\bigcup_{\alpha \in a} lab^x(\alpha)$.

By Def. 7.3 we can define a LAD that is able to keep track of the logical relevance throughout the whole ontology. Before proceeding further, we want to include an example to support the understanding of what follows.

Example 2. Consider the toy ontology as in Example 1. Then, the labelling function Lab^\perp is defined as follows:

- $a_1 \mapsto \{\text{Animal}\}$,
- $a_2 \mapsto \{\text{Person}\}$,
- $a_3 \mapsto \{\text{Vegan, Person, eats}\}$,
- $a_4 \mapsto \{\text{Student}\}$,
- $a_5 \mapsto \{\text{GraduateStudent, Student, hasHabitat}\}$,
- $a_6 \mapsto \{\text{Car}\}$,
- $a_7 \mapsto \{\text{Truck}\}$,
- $a_8 \mapsto \{\text{Car, Truck}\}$.

The corresponding \perp -LAD is represented in Fig. 1. The terms `hasGender`, `hasHabitat`,

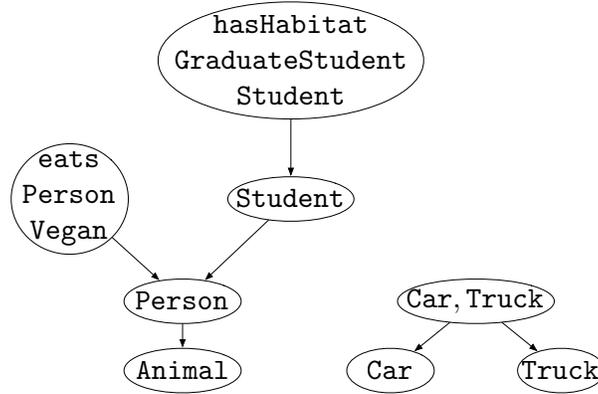


Figure 1. \perp -LAD of our example ontology

Vegetable, Mushroom, University, BA, and BS are not shown in this LAD, since they are not constrained from the above in this ontology.

In order to obtain functions that map each term to the set of logically related *atoms*, rather than axioms, we can invert the labelling functions just defined.

Definition 8. Given a notion $x \in \{\perp, \top\}$, we define the set of home atoms of t to be $h_{\mathcal{O}}^x(t) = \{a \mid \exists \alpha \in a, t \in lab^x(\alpha)\}$.

As an example, we then have that in our toy ontology $\mathbf{h}_{\mathcal{O}}^{\perp}(\text{Student}) = \{\mathbf{a}_5, \mathbf{a}_6\}$.

Definition 9. Given an ontology \mathcal{O} , a notion of locality $x \in \{\perp, \top\}$, the x -LAD of \mathcal{O} $(\mathcal{A}(\mathcal{O}), \succ, \text{Lab}^x)$, and a term $\mathfrak{t} \in \tilde{\mathcal{O}}$, we define the double cone of x -relevance for \mathfrak{t} to be the set

$$\mathbf{DC}_{\mathcal{O}}^x(\mathfrak{t}) = \bigcup_{\mathbf{a} \in \mathbf{h}_{\mathcal{O}}^x(\mathfrak{t})} (\downarrow \mathbf{a} \cup \uparrow \mathbf{a}).$$

We conjecture that the LADs described in this paper allows us to identify which axioms are logically relevant for a term in an ontology.

Conjecture 2. If an axiom α is relevant for a term \mathfrak{t} , then $\alpha \in \mathbf{DC}_{\mathcal{O}}^{\perp}(\mathfrak{t}) \cup \mathbf{DC}_{\mathcal{O}}^{\top}(\mathfrak{t})$.

6 A Consequence of ADs on Models

Finally, we want to look closer at the inseparability relation between two ontologies $\mathcal{O}_1 \subsetneq \mathcal{O}_2$ such that $\mathcal{O}_1 \equiv_{\mathcal{O}}^{mCE} \mathcal{O}_2$. Then, we have by definition that:

$$\{\mathcal{I} \mid \mathcal{I} \models \mathcal{O}_1\} = \{\mathcal{J} \mid_{\mathcal{O}_1} \mathcal{J} \models \mathcal{O}_2\}.$$

Intuitively, we can think of \mathcal{O}_2 as extending \mathcal{O}_1 *without spoiling the models* already identified for \mathcal{O}_1 . This notion can be formalised as in what follows.

Definition 10. A chain of mCEs in \mathcal{O} is a family of ontologies $\mathcal{O}_1 \subsetneq \dots \subsetneq \mathcal{O}_{\ell} = \mathcal{O}$ such that $\mathcal{O}_i \equiv_{\mathcal{O}_i}^{mCE} \mathcal{O}_{i+1}$ for $i \in \{1, \dots, \ell\}$.

Our aim is to identify a chain of mCEs in an ontology \mathcal{O} by using a x -AD. In the following proposition we are going to use the notion of a *join* $\vee(\mathbf{a}_1, \dots, \mathbf{a}_{\kappa})$ of κ atoms, defined as the minimal module that contains all the atoms in $\{\mathbf{a}_1, \dots, \mathbf{a}_{\kappa}\}$.

Conjecture 3. Let \mathcal{O} be an ontology, $(\mathcal{A}(\mathcal{O}), \succ)$ be its x -AD, with $x \in \{\perp, \top\}$. Then, each chain of ontologies $\mathcal{O}_1 \subsetneq \dots \subsetneq \mathcal{O}_{\ell} = \mathcal{O}$ that respects the following criteria is a chain of mCEs in \mathcal{O} .

1. if $\mathcal{O}_j \supseteq \mathbf{a}$ and $\mathbf{a} \succ \mathbf{b}$, then $\mathcal{O}_j \supseteq \mathbf{b}$
2. if $\mathcal{O}_j \supseteq \mathbf{a} \cup \mathbf{b}$, then $\mathcal{O}_j \supseteq \vee(\mathbf{a}, \mathbf{b})$.

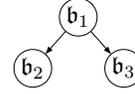
Definition 11. Let $x \in \{\perp, \top\}$ be a notion of module and \mathcal{O} be an ontology. Then, a chain $\mathcal{O}_1 \subsetneq \dots \subsetneq \mathcal{O}_{\ell} = \mathcal{O}$ of mCEs in \mathcal{O} defined via the x -LAD of \mathcal{O} is called x -chain of mCEs in \mathcal{O} .

To make the discussion clearer, let us consider the \perp -LAD as in Fig. 1 and refer to Example 1 for the axioms in each atom. Set $\mathcal{O}_1 = \{\alpha_7\}$, $\mathcal{O}_2 = \{\alpha_7, \alpha_8\}$, and $\mathcal{O}_3 = \{\alpha_7, \alpha_8, \alpha_9\}$. Then, $\mathcal{O}_1 \subsetneq \mathcal{O}_3 \subsetneq \mathcal{O}$ is a chain of mCEs, whilst $\mathcal{O}_1 \subsetneq \mathcal{O}_2 \subsetneq \mathcal{O}$ is not. Moreover, if we want to preserve everything that constrains the term *Person* from above, we see that we have to consider the principal ideal $\downarrow \mathbf{a}_3$.

Please note that for different ontologies we can still have that including the join of some set of atoms is not necessary to have a chain of mCEs, as described in the following example.

Example 3. Consider the following ontology \mathcal{O}' and its \perp -LAD:

$(\beta_1) \text{Bicycle} \sqsubseteq \text{NonMotorVehicle} \sqcap \text{TwoWheelsVehicle}$,
 $(\beta_2) \text{NonMotorVehicle} \sqsubseteq \neg \exists \text{hasPart.Engine}$,
 $(\beta_3) \text{TwoWheelsVehicle} \sqsubseteq = 2 \text{hasWheel.Wheel}$.



We see that the \perp -AD of this ontology consists of 3 atoms: $\mathfrak{b}_i = \{\beta_i\}$ for $i = 1, 2, 3$, and the inherited poset structure is $\mathfrak{b}_1 \succ \mathfrak{b}_2$, $\mathfrak{b}_1 \succ \mathfrak{b}_3$. In this case, the following is a chain of mCEs: $\mathcal{O}'_1 = \{\beta_2\} \subsetneq \mathcal{O}'_2 = \{\beta_2, \beta_3\} \subsetneq \mathcal{O}'$, even if \mathcal{O}'_2 does not contain the join $\vee(\mathfrak{b}_2, \mathfrak{b}_3)$.

7 Conclusion and Future Work

In this paper we have introduced a notion of logical relevance in ontologies. Moreover, we have shown a promising way to reveal the relevance relations between the axioms and the terms of an ontology with a suitable LAD. Finally, we have described an interesting conjecture relating the models of an ontology and the LAD of an ontology.

Future work are 4-fold, and include:

1. the completion of the theoretical investigation, that misses the proofs for many conjectured results.
2. an experimental analysis of how on average the double cone of relevance of a term spans across an ontology. The experiment will take as an input some large datasets of diverse ontologies, for example the NCBO BioPortal ontology repository.
3. an experimental comparison between the notions $\mathbf{C}_{\mathcal{O}}^{\perp}$ and $\mathbf{C}_{\mathcal{O}}^{\top}$ of direct relevance based on LADs, and some frame-based notions, as the description and the usage views in Protégé 4. Chances are that these sets will not differ much. However, such a result would provide the frame-based views with a semantic foundation. Moreover, we will be able to analyze which kinds of logically related axioms are missed, or logically unrelated axioms are included, when we rely on a syntax-based approach to logical relevance.
4. an investigation on possible applications of the notions introduced in this paper. A preliminary idea consists of considering x -chains of mCEs when it comes to reason over an ontology. In fact, we know that, for any non-empty \perp -module \mathcal{M} , and for any concept $A \in \widetilde{\mathcal{M}}$, then $\widetilde{\mathcal{M}}$ contains also all the subsumees of A . This means that we can use the \perp -LAD to predict which concepts can be a subsumee of A .

Acknowledgements We would like to thank the anonymous referees and our colleagues I. Palmisano, E. Mikroyannidi, D. Tsarkov, and N. Matentzoglou for their comments and suggestions, which helped in improving this paper considerably.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. F. (eds.) The description logic handbook: Theory, implementation, and applications. Cambridge University Press. (2003)

2. Baader, F., Bienvenu, M., Lutz, C., and Wolter, F.: Query answering over DL ABoxes: How to pick the relevant symbols. In Cuenca Grau, B., Horrocks, I., Motik, B., and Sattler, U. (eds.): Proc. of DL-09. *ceur-ws.org*, vol. 477. (2009)
3. Cuenca Grau, B., Horrocks, I., Kazakov, Y., and Sattler, U.: Modular reuse of ontologies: Theory and practice. In: *J. of Artif. Intell. Research*, vol. 31, pp. 273–318. (2008)
4. Del Vescovo, C.: The modular structure of an ontology: Atomic decomposition towards applications. In Rosati, R., Rudolph, S., and Zakharyashev, M. (eds.): Proc. of DL-11. *ceur-ws.org*, vol. 745. (2011)
5. Del Vescovo, C., Gessler, D., Klinov, P., Parsia, B., Sattler, U., Schneider, T., and Winget, A.: Decomposition and modular structure of BioPortal ontologies. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., and Blomqvist, E. (eds.): Proc. of ISWC-11. (2011)
6. Del Vescovo, C., Parsia, B., Sattler, U., and Schneider, T.: The modular structure of an ontology: Atomic decomposition. Walsh, T. (eds.): Proc. of IJCAI-11. (2011)
7. Del Vescovo, C., Parsia, B., and Sattler, U.: Topicality in logic-based ontologies. Andrews, S., Polovina, S., Hill, R., and Akhgar, B. (eds.): Proc. of ICCS-11. (2011)
8. S. Ghilardi, Lutz, C., and Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In Doherty, P., Mylopoulos, J., and Welty, C. (eds.): Proc. of KR-06 pp. 187–197. AAAI Press. (2006)
9. Jiménez-Ruiz, E., Cuenca Grau, B., Sattler, U., Schneider, T., and Berlanga Llavori, R.: Safe and economic re-use of ontologies: A logic-based methodology and tool support. In: Proc. of ESWC-08. LNCS, vol. 5021, pp. 185–199. (2008)
10. Jimeno, A., Jiménez-Ruiz, E., Berlanga Llavori, R., and Rebholz-Schuhmann, D.: Use of shared lexical resources for efficient ontological engineering. In: SWAT4LS-08. *ceur-ws.org*. (2008)
11. Konev, B., Lutz, C., Ponomaryov, D., and Wolter, F.: Decomposing description logic ontologies. In: Proc. of KR-10. pp. 236–246. (2010)
12. Konev, B., Lutz, C., Walther, D., and Wolter, F.: Formal properties of modularization. In: *Modular Ontologies*, LNCS, vol. 5445, pp. 25–66. Springer-Verlag. (2009)
13. Kontchakov, R., Pulina, L., Sattler, U., Schneider, T., Selmer, P., Wolter, F., and Zakharyashev, M.: Minimal module extraction from DL-Lite ontologies using QBF solvers. In: Proc. of IJCAI-09. pp. 836–841. (2009)
14. Parsia, B., and Schneider, T.: The modular structure of an ontology: An empirical study. In: Proc. of KR-10. pp. 584–586. AAAI Press (2010)
15. Sattler, U., Schneider, T., and Zakharyashev, M.: Which kind of module should I extract? In Cuenca Grau, B., Horrocks, I., Motik, B., and Sattler, U. (eds.): Proc. of DL-09. *ceur-ws.org*, vol. 477. (2009)

Experimental Results on Solving the Projection Problem in Action Formalisms Based on Description Logics

Wael Yehia,¹ Hongkai Liu,² Marcel Lippmann,² Franz Baader,² and Mikhail Soutchanski³

¹ York University, Canada

w2yehia@cse.yorku.ca

² TU Dresden, Germany

lastname@tcs.inf.tu-dresden.de

³ Ryerson University, Canada

mes@scs.ryerson.ca

Abstract. In the reasoning about actions community, one of the most basic reasoning problems is the projection problem: the question whether a certain assertion holds after executing a sequence of actions. While undecidable for general action theories based on the situation calculus, the projection problem was shown to be decidable in two different restrictions of the situation calculus to theories formulated using description logics.

In this paper, we compare our implementations of projection procedures for these two approaches on random testing data for several realistic application domains. Important contributions of this work are not only the obtained experimental results, but also the approach for generating test cases. By using patterns extracted from the respective application domains, we ensure that the randomly generated input data make sense and are not inconsistent.

1 Introduction

The situation calculus (SC) [7] is a popular many-sorted language for representing action theories. The *projection problem* is one of the most basic reasoning problems for action formalisms such as the SC. It deals with the question whether a certain formula holds after executing a sequence of actions from an initial state. Since the situation calculus encompasses full first-order logic, the projection problem for action theories formulated with it is in general undecidable. One possibility to restrict SC such that the projection problem becomes decidable is to use a decidable fragment of first-order logic instead of full first-order logic as underlying base logic [3, 5, 8]. Description Logics (DLs) [1] are well-suited for this purpose since they are well-investigated decidable fragments of first-order logic that have been used as knowledge representation formalisms in various application domains.

For the DL-based action formalism introduced in [3], it could indeed be shown that, in this restricted setting, the projection problem is decidable. Basically, the procedure for solving the projection problem developed in [3] works as follows. Using time-stamped copies of all relevant concept and role names in the input, one can describe the initial state and the subsequent states (obtained by executing the actions of a given sequence of actions one after the other) in an ABox and an (acyclic) TBox. Solving the projection problem is thus reduced to checking whether an appropriately time-stamped version of the assertion to be checked is entailed by this ABox w.r.t. this TBox.

For the SC-based action formalism introduced in [5, 8], the projection problem is solved by *regression*, as usual in situation calculus. In this approach, one transforms a query formula and an action sequence into a regressed formula. It is then checked whether the regressed formula is entailed from an incomplete description of the initial state and the unique name axioms. To obtain decidability, it is thus not sufficient to restrict the base logic (in which the initial state, the axioms, and the query formula are written) to a decidable fragment of first-order logic. One must also show, as done in [5, 8], that regression can be realised within this fragment, i.e., that the regressed formula belongs to this fragment.

Both approaches for solving the projection problem have been implemented. In this paper, we compare these implementations on random testing data for several realistic application domains. In addition to describing the obtained experimental results, this paper also sketches our approach for generating the test cases. By using typical patterns extracted from the respective application domains, we ensure that the randomly generated input data make sense and are not inconsistent.

The rest of the paper is organised as follows. After a short presentation of the two action formalisms and their respective approaches for solving the projection problem, the test case set-up is explained and the experimental results are presented. The paper concludes with a discussion of the results obtained and possible future work.

2 DL-Based Action Formalisms

In this section, we explain the two approaches for deciding the projection problem on an intuitive level. More details on the directly DL-based action formalism can be found in [3] and on the SC-based action formalism restricted to DLs in [5, 8]. We also discuss the differences between the two approaches.

We assume the reader to be familiar with basic notions of Description Logics (DLs), which can e.g. be found in [1]. In the following, we use action formalisms that are based on the Description Logic \mathcal{ALCO}^U , which extends the basic DL \mathcal{ALC} with nominals, i.e., concepts interpreted as singleton sets (letter \mathcal{O}), and the universal role, which is interpreted as the binary relation that links any two domain elements (superscript U). In the following, the universal role will be denoted by the letter U .

Formulated for DL-based action formalisms, the *projection problem* deals with the following question: Given an initial ABox describing the initial state, a TBox describing the domain constraints, and a sequence of actions, does a query, i.e., an assertion, hold after executing the action sequence from the initial state? In the following, we call the approach for solving the projection problem introduced in [3] the *reduction approach* and the approach for solving the progression problem introduced in [5, 8] the *regression approach*.

In this paper, we will use examples from the *logistics domain* [4]. The domain represents a simplified version of logistics planning. Various objects, e.g. boxes, packages and luggage, can be transported by vehicles, such as airplanes and trucks, from one location to another. Locations, such as airports and warehouses, are located in cities.

2.1 The Reduction Approach

The action formalism [3] to which this approach applies, describes the possible initial states by an \mathcal{ALCO}^U -ABox and the domain constraints by an acyclic TBox. The restriction to acyclic TBoxes avoids anomalies caused by the so-called ramification problem (see [2, 6] for approaches that can deal with more general TBoxes). It remains to describe how actions are formalised. In contrast to the general setting introduced in [3],⁴ the actions used here are without occlusions,⁴ which can be omitted since they are not required for the application domains used in our experiments. Basically, an action consists of a set of pre-conditions and a set of post-conditions. Both are given as ABox assertions. Pre-conditions must be satisfied for the action to execute and post-conditions describe what new assertions must be satisfied after the application of the action. Post-conditions can be conditional, i.e., the required change is only made in case the condition is satisfied. The following action, for instance, causes the airplane `OurBoeing777` to fly from `AirportJFK` to `AirportSIN` transporting `Box42`, which is the only cargo:

$$\begin{aligned} \text{fly}_1 := & (\{\text{at}(\text{OurBoeing777}, \text{AirportJFK})\}, \\ & \{\{\text{loaded}(\text{Box42}, \text{OurBoeing777})\}/\neg\text{at}(\text{Box42}, \text{NYC}), \\ & \{\text{loaded}(\text{Box42}, \text{OurBoeing777})\}/\text{at}(\text{Box42}, \text{Singapore})\}). \end{aligned}$$

The pre-condition $\text{at}(\text{OurBoeing777}, \text{AirportJFK})$ ensures that this action is only applicable if `OurBoeing777` is indeed at `AirportJFK`. The effect of this action, described by two conditional post-conditions, is that the box `Box42` is no longer at `NYC` but at `Singapore` if it was loaded to `OurBoeing777`. This action is a simplified version of an action used in our experiments.

According to the semantics of DL actions defined in [3], nothing should change that is not explicitly required to change by some post-condition. It was shown in [3] that this action formalism can be seen as an instance of Reiter's situation calculus, i.e., there is a translation to SC.

⁴ Occlusions describe which parts of the domain can change arbitrarily when an action is applied. Details about occlusions can be found in [3].

In [3], the projection problem for this action formalism is reduced to the consistency problem of an ABox w.r.t. an acyclic TBox. The central idea of the reduction is to create time-stamped copies of all concept, role, and individual names. Intuitively, A^0 is the concept A before executing any action, A^1 after executing the first action, A^2 , after subsequently executing the second one, etc. Using such time-stamped copies, the effect that executing the actions has on named individuals, i.e., domain elements that correspond to an individual name, can be described using ABox assertions. Since post-conditions only state changes for named individuals, nothing should change for unnamed ones. This is expressed using an acyclic TBox.

Note that the reduction in [3] does not deal with the universal role, but integrating it into the reduction is not hard. In fact, its interpretation never changes, and it is not allowed to occur in post-conditions. We also applied some simple optimisations to the original reduction, which however proved to be quite valuable w.r.t. the time and memory consumption of our implementation.

2.2 The Regression Approach

This approach is based on Basic Action Theories (BATs) in the Situation Calculus (SC) [7]. In general, a BAT consists of pre-condition axioms (PAs) and successor state axioms (SSAs), which describe the effects and non-effects of actions; an initial theory, which describe the possible initial states; an situation independent acyclic terminology describing convenient definitions of the application domain; a set of domain independent foundational axioms; and axioms for the unique-name assumption (UNA). In SC, a sequence of actions is called a *situation*, and the empty sequence the *initial situation*. Applying actions can change the interpretations of certain predicates, which are called *fluents*. Fluents have an additional argument position to express the situation in which the fluent is considered. Given the initial theory, the situation determines which are the possible current states, i.e., how the fluents have been changed by executing the sequence of actions. A fundamental problem in reasoning about actions is the frame problem, i.e., how one compactly represents numerous non-effects of actions. Reiter's approach [7] for solving this problem is based on quantifying over action variables in SSAs. This approach is adapted to the context of BATs based on decidable description logics in [5, 8].

In the context of BATs in SC, *regression* is the act of converting the query formula (which should hold after executing the action sequence) to a formula that should hold in the initial situation by making use of the SSAs and the domain constraints. In the regression approach used in this paper, solving the projection problem in certain BATs based on a restricted first-order language, called \mathcal{L} , is reduced to solving the satisfiability problem in the Description Logic \mathcal{ALCO}^U . This is done by imposing precise syntactic constraints on Reiter's SSAs to guarantee that after doing the logically equivalent transformations required by regression, the resulting regressed formula remains in \mathcal{L} , if the projection query formula is in \mathcal{L} . Instead of defining this approach in detail, which is not possible

due to space constraints, we illustrate it on an example. Detailed definitions can be found in [5, 8].

There is a PA axiom for each action that describes the pre-conditions for executing the action. For instance, the PA of the action fly looks as follows:

$$\text{Poss}(\text{fly}(z_1, z_2, z_3), S) = \text{airplane}(z_1) \wedge \text{airport}(z_2) \wedge \text{airport}(z_3) \wedge \\ \text{at}(z_1, z_2, S) \wedge (z_2 \neq z_3)$$

Similarly, there is one SSA per fluent that describes the conditions under which actions make the fluent true or false. This is the SSA of the fluent loaded:

$$\text{loaded}(x, y, \text{do}(a, S)) = (\exists z_1. a = \text{load}(x, y, z_1) \wedge \text{ready}(x, S)) \vee \\ (\text{loaded}(x, y, S) \wedge \neg(\exists z_1. a = \text{unload}(x, y, z_1)))$$

The domain constraints are basically acyclic TBox axioms, but written in the language \mathcal{L} . An example would be the following:

$$\text{object}(x) \equiv \text{box}(x) \vee \text{luggage}(x)$$

The initial theory and the query are \mathcal{L} sentences that are restricted such that they can be transformed into \mathcal{ALCO}^U -assertions. The following is an example of an initial theory \mathcal{D}_{S_0} and a query W .

$$\mathcal{D}_{S_0} = \text{truck}(T_1) \wedge \text{box}(B_1) \wedge \text{location}(L_1) \wedge \forall x. (\text{box}(x) \supset \text{loaded}(x, T_1)) \\ W = \text{loaded}(B_1, T_1, \text{do}([\text{load}(B_1, T_1, L_1), \text{unload}(B_1, T_2, L_1)], S_0))$$

Note that the action sequence is here viewed to be part of the query, whereas for the reduction approach it is given separately.

2.3 Comparison of the Two Approaches and Restrictions

Both approaches solve the projection problem for an action formalism that is based on the DL \mathcal{ALCO}^U . The action sequence is in both cases a list of ground actions, i.e., they do not contain free variables. However, the actions considered in the regression approach are more general than the ones in the reduction approach. In fact, the actions of the latter approach are so-called local-effect actions, which can only effect changes for *named* individual (i.e., ones that are explicitly named by a constant symbol), whereas global-effect actions, which are allowed in the former approach, can also effect changes for unnamed individuals. For example, the action fly in the regression approach moves all objects loaded to the airplane from one city to the other, independent of whether these objects have a name or not. Its approximation in the reduction approach moves only objects that are explicitly mentioned by their name in the description of the action. Though this approximation may in principle lead to different answers for the projection problem, this never happened in our experiments.

Another difference are the languages in which the queries are formulated. In the reduction approach, only instance queries are considered, whereas the other

approach formulates queries in a restricted first-order language, with explicit variables and (unguarded) quantification. However, for the queries considered in our experiments, we were able to use the universal role to bridge this gap.

Regarding the complexity of the projection problem, regression may take up to double exponential time and space, whereas the reduction approach takes polynomial space. However, this is the worst-case complexity of the approaches. The more expressive regression approach is assumed to hit this bound only in non-practical cases.

In the next section, we show how these approaches behave in practice by evaluating them on randomly generated input data that are modelled on patterns occurring in applications.

3 Test Case Generation and Experimental Results

In order to evaluate the two approaches for solving the projection problem, we generated testing data inspired by several application domains. Of course, the representation of the input data is different for the two approaches. Basically, we generated the data for the regression approach, and then translated them into the format required by the reduction approach. As already mentioned, global-effect actions are approximated by local effect actions that make changes to all the named individuals. Otherwise, the translation preserves the semantics of the action theory. To increase readability for DL-literate readers, we describe the translated input data rather than the SC-based ones. Also, we use the size of the translated input data as size of the input.

In general, a test case is divided into a fixed part and a varying part. The fixed part consists of descriptions of all available actions and an acyclic TBox describing the domain. The varying part consisting of the initial ABox, the query to be asked, and the action sequence.

Subsequently, we sketch one domain, the *logistics domain*, to give an impression about what kind of projection problems can be expressed and solved by the formalisms.⁵ The examples used above already gave an impression of how the domain looks like. It represents a simplified version of logistics planning. Various objects, e.g. boxes and luggage, can be transported by vehicles, such as airplanes and trucks, from one location to another.

More precisely, the fixed part is as follows. The acyclic TBox describes general facts about the domain. It contains the following axioms:

$$\begin{aligned} \text{Object} &\equiv \text{Box} \sqcup \text{Luggage} \\ \text{Vehicle} &\equiv \text{Truck} \sqcup \text{Airplane} \\ \text{Truck} &\equiv \text{TransportTruck} \sqcup \text{MailTruck} \sqcup \text{RecyclingTruck} \\ \text{Location} &\equiv \text{Airport} \sqcup \text{Garage} \sqcup \text{Street} \sqcup \text{Warehouse} \end{aligned}$$

⁵ See http://www.cse.yorku.ca/~w2yehia/dl2012_results.html for further domains.

The initial ABox contains incomplete knowledge about the initial state. It states static facts like $\text{Airplane}(\text{OurBoeing777})$, $\text{Airport}(\text{AirportJFK})$, $\text{Box}(\text{Box42})$, $\text{Truck}(\text{OurTruck})$, $\text{City}(\text{NYC})$, $\text{inCity}(\text{OurGarage}, \text{NYC})$, etc. In addition, the initial ABox contains dynamical knowledge like $\text{at}(\text{OurTruck}, \text{AirportJFK})$, $\text{Ready}(\text{Box42})$, $\text{loaded}(\text{Box127}, \text{OurBoeing777})$, which means that OurTruck is at AirportJFK , Box42 is ready to be loaded, and Box127 is loaded into OurBoeing777 .

The action sequences are built using the atomic actions load , unload , fly , and driveTruck , instantiated with appropriate individuals. The action load loads an object into a vehicle at a location, e.g. $\text{load}(\text{Box42}, \text{OurBoeing777}, \text{AirportJFK})$ is an instance of this action. It is executable if both Box42 and OurBoeing777 are at AirportJFK , i.e., $\text{at}(\text{Box42}, \text{AirportJFK})$ and $\text{at}(\text{OurBoeing777}, \text{AirportJFK})$ hold, and Box42 is ready to be loaded, i.e., we have $\text{Ready}(\text{Box42})$. The action unload is defined analogously for unloading an object from a vehicle at a location. Executing the action fly means that an airplane flies from one airport to another one, e.g. $\text{fly}(\text{OurBoeing777}, \text{AirportJFK}, \text{AirportSIN})$. This action is executable if we have $\text{at}(\text{OurBoeing777}, \text{AirportJFK})$, and the two airports are not the same, which is the case in the example. The action driveTruck is similar. It is used to drive a truck from one location to another one. An example for such an action would be $\text{driveTruck}(\text{OurTruck}, \text{OurGarage}, \text{AirportJFK})$. Executing the action is only possible if OurGarage and AirportJFK are at the same city.

On the DL side, queries are instance queries, i.e., concept assertions of the form $C(a)$ where C is a \mathcal{ALCO}^U -concept description built using the concept names and role names introduced above and a is an individual name such as Box42 , AirportJFK , OurBoeing777 , etc.

To properly test our implementations, we obviously need some sort of automated generation of input data, but due to the non-trivial expressivity of the language at hand, it is hard to generate useful test cases. We could not find any precedence for such an attempt, i.e., generating random projection problem test cases, in the literature. For planning domains [4] there are some people working on test case generation, but they are dealing with STRIPS or some of its extensions, which are mostly at the propositional level. For such inexpressive formalisms, it is easier to generate input data that make sense. Generating purely random \mathcal{ALCO}^U -ABoxes and instance queries usually yields meaningless queries or initial theories that are inconsistent or action sequences that are not executable.

We describe now more precisely how we generate the testing data. The TBox contains the axioms above plus some auxiliary axioms described below. The initial ABox contains both static knowledge and dynamical knowledge. For the static knowledge, it makes no sense to add incompleteness, because usually we know, for instance, that Box42 is a box. We omit, however, axioms stating facts like Box42 is not an airplane, i.e., $\neg\text{Airplane}(\text{Box42})$. The real incompleteness concerns the dynamical knowledge. We found it useful to generate more or less complete knowledge by generating a number of boxes, trucks, airplanes, airports, and cities together with static knowledge like in which city airports are located. We generate also assertions for the role names at , loaded and Ready

for all individuals of the respective type. We add more incompleteness by removing assertions from the ABox and adding new assertions using the following transformation rules: a \sqcup -rule, and an \exists -rule. Applying the \sqcup -rule means to take two assertions $A(a)$, $B(a)$ and replacing them with the assertion $(A \sqcup B)(a)$. For reasons of simplicity and interchangeability of the formats read by our implementations, we introduce a new concept name Aux , and add the concept definition $Aux \equiv A \sqcup B$ to the TBox, and the assertion $Aux(a)$ to the ABox.⁶ Applying the \exists -rule means to take a role assertion $r(a, b)$ and to replace it with $(\exists r. \top)(a)$. For example, $at(\text{Box127}, \text{AirportSIN})$ could be replaced with $(\exists at. \top)(\text{Box127})$. Instead of knowing that Box127 is at AirportSIN , we now just know that Box127 is somewhere, i.e., at a location. Again, we introduce a new concept name for $\exists at. \top$, and add its definition to the TBox. Of course, it makes no sense to apply the rules too often since then the resulting ABox would be too incomplete. For this reason, we use them rather sparingly: roughly there are ten-times more individuals than the number of times these rules are applied.

The query and the action sequence are generated using (domain-specific) patterns. Building formulas from hand-made patterns occurring in real applications is one step towards generating realistic test cases, but it suffers from the fact that the generated formulas might be too similar, and thus the test cases do not provide the extensive coverage that random formula generation does. Thus, we mixed patterns with a bit of randomness. We developed ten patterns for the logistics domain. Some of these patterns can take input, and generate a random input if none is given. For example, a pattern might describe whether there are any boxes on a truck in some location. The input could be any combination of box, truck or location constants. If necessary, a missing constant can be randomly generated, or it could remain as a variable otherwise. The following formula Φ was generated from such a pattern, where the input was box_i and location_j . The generated formula is translated to the \mathcal{ALCO}^U -assertion α with aux being a dummy individual.

$$\begin{aligned}\Phi &= \exists y. (\text{loaded}(\text{box}_i, y) \wedge \text{truck}(y) \wedge \text{at}(y, \text{location}_j)) \\ \alpha &= (\exists U. (\{\text{box}_i\} \sqcap \exists \text{loaded}. (\text{Truck} \sqcap \exists at. \{\text{location}_j\}))) (\text{aux})\end{aligned}$$

We use those simple but versatile patterns to generate formulas that replace the propositions in a randomly generated satisfiable formula of propositional logic in disjunctive normal form in order to obtain the query. Using this approach, it is ensured that we have more randomness at the propositional level and less randomness at the FOL level.

Finally, to generate random but executable action sequences, we used patterns again. But now a pattern is a generic description of a sequence of actions necessary to satisfy a goal. For instance, one action sequence in the logistics domain describes the process of gathering all known boxes in a particular city and transporting them to another city. The choice of the cities is random, and

⁶ Thus, the TBox contains also abbreviations for concepts, where the abbreviation occurs only once in the ABox.

picking the transportation vehicle is random as well. On top of that, we compute this action sequence based on a random initial ABox. Of course, we could have tried to pick purely random actions, but then most of the generated action sequences would not have been executable (i.e., not all pre-conditions would be satisfied). Testing randomly generated action sequences for executability takes time and would result in having to throw away most of the generated sequences.

We now present the testing results we obtained for the logistics domain. The results were obtained on an Intel® Core™ 2 Duo E8400 CPU with a clock frequency of 3.00 GHz, and 4 GB of RAM. We used JVM version 1.7.0. Both implementations use HerMiT 1.3.6 as underlying DL reasoner.

To make the results better comparable, we generated a couple of initial theories, and then manually removed individuals from them to create new smaller initial theories. Then, we generated a set of queries and action sequences for each initial theory. This way, we can measure the running time for solving the projection problem as the ABox varies in size, while the query and action sequence stay the same. Some of the testing results for about 700 test cases that we obtained can be found in Tab. 1. These results give an impression of the performance of the two approaches. We sketched how the action sequences and the queries look like. Empty cells in Tab. 1 mean that the program could not finish within two minutes, which was our cut-off time. To be useful in practice, an answer within two minutes is preferred. For the regression approach, we also measured the time for computing the regressed formula only, which is usually quite small (about 0.01 s). Thus, the main time for the regression approach is used for checking the consistency of the generated knowledge base using HerMiT. One can observe that the running times very much depend on the action sequence and the query that is asked. For action sequences of length 5 or more, both approaches seem to be struggling due to overhead taken by HerMiT. Thus, improving DL-systems will improve the running time of our approaches significantly. Note that the last test case in Tab. 1 describes a rather simple scenario, which is still too complex for both approaches. One can observe also that the size of the initial ABox affects the running time. Note that the length of the regressed formula is not affected by these variations. Also, in most cases, a query involving a value restriction on U causes a longer running time for the reduction approach.

All in all, the testing done in this paper should be considered as a first step. More testing and a careful inspection of the structure of the input is needed to give an answer to the question which approach performs better on which inputs.

4 Conclusion and Future Work

We have compared the implementations of two approaches for solving the projection problem for DL-based action formalisms. The main result of this comparison is twofold. First, it is clear that both approaches seem to be struggling when dealing with action sequences longer than five. It turned out that computing the regression formula can be done very quickly, and most of the time for the regression approach is consumed by the DL-reasoner. For the reduction approach, it is

Table 1. Testing results for the logistics domain

Action sequence/ query	Number of individuals in ABox	Time for reduction approach	Time for regression approach
load	12	0.57 s	8.51 s
$(\exists U.(\exists \text{loaded}.\{\exists \text{at}.\{\ell_1\} \sqcap \exists \text{at}.\{\ell_2\} \sqcap$ $\text{Truck}) \sqcap \text{Box}))(\text{aux})$	13	0.58 s	17.07 s
	14	0.58 s	34.51 s
	15	0.69 s	68.98 s
	16	0.59 s	—
	17	0.59 s	—
	40	0.64 s	—
driveTruck	7	0.64 s	1.36 s
$(\exists U.(\exists \text{loaded}.\{\exists \text{at}.\{\ell\} \sqcap \text{Truck}) \sqcap$ $\text{Box}))(\text{aux})$	8	0.58 s	8.50 s
	9	0.82 s	2.14 s
	10	0.55 s	4.75 s
	11	0.55 s	12.09 s
	12	0.56 s	15.15 s
	13	0.90 s	16.21 s
fly, fly	13	2.24 s	1.6 s
$(\forall U.(\neg(\text{Box} \sqcap \exists \text{at}.\{\ell\}) \sqcup \text{Ready}))(\text{aux})$			
unload, load, unload	13	0.92 s	1.69 s
$(\exists U.(\text{Box} \sqcap \exists \text{loaded}.\text{Truck}))(\text{aux})$	14	1.04 s	1.68 s
	15	0.96 s	1.64 s
unload, load, fly	12	1.75 s	4.66 s
$(\exists U.(\{b\} \sqcap \exists \text{loaded}.\text{Truck}))(\text{aux})$	13	1.76 s	8.51 s
	14	1.85 s	18.57 s
	34	6.63 s	2.72 s
load, load, load, fly	30	—	70.98 s
$(\forall U.(\neg(\exists \text{at}.\{\ell\} \sqcap \text{Box}) \sqcup$ $(\exists \text{loaded}.\{a\}))) (\text{aux})$			
unload, load, load, unload, unload	8	—	23.04 s
$(\forall U.(\neg(\exists \text{at}.\{\ell\}) \sqcap \text{Box}) \sqcup \text{Ready})(\text{aux})$	9	0.56 s	—
	10	0.57 s	—
	11	—	—
	12	—	—
unload, load, load, load, fly, unload, unload, unload	26	—	—
$(\exists U.(\{b\} \sqcap \exists \text{loaded}.\text{Truck}))(\text{aux})$			

also clear that long action sequences cannot be handled efficiently since, in the worst case, the initial ABox is copied for each point in time. Also, we observed that the length of the action sequence is not the only metric that affects the running time. The size of the initial ABox as well as the form of the query are also important. We expect that more testing will give a more accurate description on what what are the weak points of the approaches. Maybe also the actions, i.e., the size and structure of the pre-conditions and the post-conditions, need to be considered in more detail. For the reduction approach, the pre-conditions affect the runtime in our experiments, while for regression they do not. As a result of these investigations, we hope to locate the parts of the implementations that need to be optimised.

The second main result is that we gained some experience in automatically generating testing data for the projection problem. It is clear that generating good test cases is not a trivial task (comparable to generating good test cases in programming languages). Randomly generating test cases on a purely syntactic level without taking the semantics of action formalisms into account is problematic since the resulting initial ABoxes can easily be inconsistent, the action sequences not executable, etc. In these cases, projection does not make sense. To take the semantics into account, we followed a domain specific approach, and used patterns found in applications to generate data that make sense from a semantic point of view. Nevertheless, these are only first step towards semantically well-founded test case generation. More versatile approaches need to be developed as future work.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press (2003)
2. Baader, F., Lippmann, M., Liu, H.: Using causal relationships to deal with the ramification problem in action formalisms based on description logics. In: *Proc. of the 17th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-17)*. Lecture Notes in Computer Science, vol. 6397, pp. 82–96. Springer-Verlag (2010)
3. Baader, F., Lutz, C., Miličić, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI-05)*. AAAI Press (2005)
4. Bacchus, F.: The AIPS '00 planning competition. *AI Magazine* 22(3), 47–56 (2001)
5. Gu, Y., Soutchanski, M.: A description logic based situation calculus. *Ann. Math. Artif. Intell.* 58(1-2), 3–83 (2010)
6. Lin, F., Soutchanski, M.: Causal theories of actions revisited. In: *Proc. of the 25th AAAI Conf. on Artificial Intelligence (AAAI-2011)*. pp. 235–240 (2011)
7. Reiter, R.: *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. The MIT Press, Bradford Books, Cambridge, Massachusetts, USA (2001)
8. Soutchanski, M., Yehia, W.: Towards an expressive decidable logical action theory. In: *Proc. of the 25th Int. Workshop on Description Logics (DL-2012)* (2012)

Efficient Upper Bound Computation of Query Answers in Expressive Description Logics

Yujiao Zhou, Bernardo Cuenca Grau, and Ian Horrocks

Department of Computer Science, University of Oxford, UK

1 Introduction

In recent years, there has been a growing interest in the problem of conjunctive query answering over description logic (DL) ontologies and large scale data sets. This problem is central to many applications, which often involve managing data sets consisting of hundreds of millions, or even billions of data assertions.

Meeting the scalability requirements of such applications is, however, a very challenging problem. Answering conjunctive queries over ontologies in expressive DLs is of high computational complexity; in fact, decidability is still an open problem for *SR_{OIQ}* [14] —the DL that underpins the standard ontology language OWL 2 [6]. Although small restriction on the ontology or query language can ensure decidability [26], worst case complexity is still very high (at least 2NEXPTIME) [15]. Several OWL/*SR_{OIQ}* reasoners, including HermiT [20], FaCT++ [28] and Racer [12], support query answering for restricted classes of conjunctive queries, but in spite of intensive efforts at optimisation, they can still only deal with small to medium size data sets [17, 13].

Scalability of query answering can be ensured by restricting the expressive power of the ontology language to the level that makes reasoning tractable. This has led to the development of three profiles of OWL 2, namely OWL 2 RL, OWL 2 EL, and OWL 2 QL [21]; these profiles are based on (families of) “lightweight” description logics, notably the DLP [10], DL-Lite [4], and the \mathcal{EL} families of DLs [2], respectively. Query answering in all these lightweight DLs can be implemented in polynomial time w.r.t. the size of data (and even in LOGSPACE in the case of DL-Lite). Such appealing theoretical properties have spurred the development of specialised reasoning techniques [24, 22, 18, 4, 16].

Although allowing for efficient query answering, these lightweight DLs impose severe restrictions on the expressive power of the ontology language. In order to provide scalable query answering w.r.t. ontologies that cannot be captured by such lightweight DLs, Semantic Web researchers have developed reasoners that can process arbitrary OWL/*SR_{OIQ}* ontologies, but that guarantee completeness only for ontologies that fall within the fragment defined by the OWL 2 RL profile. Given the close connection between OWL 2 RL and datalog, these reasoners typically implement (deductive) database algorithms based on data materialisation. Examples of such systems include Jena [19] and OWLim [16].

All such materialisation-based reasoners are *sound*; that is, the answers they compute can be seen as a *lower bound* on the complete set of query answers.

For ontologies outside the OWL 2 RL profile, however, these reasoners are *incomplete*, and hence they are not guaranteed to compute all query answers. A possible approach to this problem is to investigate the behaviour of the reasoner on a given query Q and ontology \mathcal{T} in an effort to show that it behaves as a complete reasoner w.r.t. Q , \mathcal{T} and an arbitrary data set [7]. Providing such guarantees is, however, not always possible.

An alternative approach, which we investigate in this paper, is to devise a scalable procedure for computing complete but possibly unsound query answers. Such answers provide an *upper bound* to the set of all answers, which can complement the lower bounds efficiently computed by incomplete reasoners. The combined use of such lower and upper bounds has many interesting implications. First, the difference between the upper and lower bounds can be used as an optimisation for reducing the number of candidate answers; furthermore, it also provides a measure of the degree of incompleteness of a reasoner for a given input; finally, if both bounds match, we can efficiently compute all query answers without relying on potentially very expensive entailment tests.

In order to be useful, upper bounds should clearly be as tight as possible, and should also be efficiently computable. Obtaining such an upper bound is, however, not straightforward. The technique we use is to approximate \mathcal{T} to give an ontology \mathcal{T}' that is strictly “stronger” than \mathcal{T} (i.e., $\mathcal{T}' \models \mathcal{T}$), and that is within a fragment for which query answers can be efficiently computed.

Knowledge approximation has been extensively studied in the literature, although mostly in the direction of weakening the ontology/theory [8, 23]. There has also been some work on strengthening approximations. For propositional logic, Horn theories can be used to both strengthen and weaken the original theory [27]; these Horn approximations can then be used to optimise reasoning by exploiting the more efficient inference in the Horn theories. Finally, approximation is also related to the computation of least common subsumers [1].

Our technique exploits recent work on chase termination for existential rules, which introduces a so called Models-Summarising Acyclicity (MSA) check [5]. MSA is an approximation of existential rules (datalog[±]) into datalog. As most *SR_QI_Q* TBoxes can be translated into existential rules extended with disjunction (datalog^{±,∨}) using model-preserving transformations, we can adapt MSA to produce a datalog approximation of a *SR_QI_Q* TBox. Moreover, the resulting datalog rules can be translated back into an OWL 2 RL TBox for which complete query answers can be computed using materialisation based reasoners.

We have implemented our approach, and conducted preliminary experiments using LUBM [11]. Our preliminary results suggest that our bound could be tight for many queries, and it can be computed efficiently (or at least with similar efficiency to computing the lower bound).

2 Preliminaries

We assume basic familiarity with the DLs underpinning the ontology language OWL 2 and its profiles. We next introduce datalog^{±,∨} and datalog languages,

and define the syntax and semantics of conjunctive queries. In our definitions, we adopt standard first-order logic notions of variable, constant, term, substitution, atom, formula, and sentence. We assume all formulas to be function-free. We denote with \perp the special atom evaluated as false in all interpretations, and we use \approx to denote the special equality predicate in first-order logic. Finally, we also adopt the standard notions of satisfiability, unsatisfiability, and entailment.

Datalog Languages. A datalog ^{\pm, \vee} rule r is a formula of the form (1), where each B_j is an atom different from \perp whose free variables are contained in \mathbf{x} , and

- $m = 1$ and $\varphi_1(\mathbf{x}, \mathbf{y}_1) = \perp$, or
- $m \geq 1$ and, for each $1 \leq i \leq m$, formula $\varphi_i(\mathbf{x}, \mathbf{y}_i)$ is a conjunction atoms different from \perp whose free variables are contained in $\mathbf{x} \cup \mathbf{y}_i$.

$$\forall \mathbf{x}. [B_1 \wedge \dots \wedge B_n] \rightarrow \bigvee_{i=1}^m \exists \mathbf{y}_i. \varphi_i(\mathbf{x}, \mathbf{y}_i) \quad (1)$$

A rule is safe if each variable in \mathbf{x} also occurs in some B_j , and we consider only safe rules. For brevity, the quantifier $\forall \mathbf{x}$ is left implicit. The *body* of r is the set $\mathbf{body}(r) = \{B_1, \dots, B_n\}$, and the *head* of r is the formula $\mathbf{head}(r) = \bigvee_{i=1}^m \exists \mathbf{y}_i. \varphi_i(\mathbf{x}, \mathbf{y}_i)$. A datalog ^{\pm, \vee} rule r is datalog ^{\pm} if $m = 1$ [3], and it is datalog if it is datalog ^{\pm} and the head is a single atom without existential quantifiers.

A *fact* is a ground atom, and an *instance* I is a finite set of facts. We denote with $\mathbf{ind}(I)$ the set of all individuals occurring in I .

For Σ a set of datalog rules and I an instance, the *saturation* of Σ w.r.t. I is the instance I' consisting of all facts entailed by $\Sigma \cup I$.

Most DL ontologies can be transformed into a set of datalog ^{\pm, \vee} rules and an instance by means of standard transformations. The rules and facts obtained via such transformations involve only unary and binary predicates; thus, we define an ABox \mathcal{A} as an instance containing only unary and binary atoms.

Queries. A *conjunctive query* (CQ), or simply a *query*, is a formula $Q(\mathbf{x})$ of the form $\exists \mathbf{y}. \varphi(\mathbf{x}, \mathbf{y})$, where $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms. A tuple of individuals \mathbf{a} is a *certain answer* to a query $Q(\mathbf{x})$ w.r.t. a set of first-order sentences \mathcal{F} and an instance I if $\mathcal{F} \cup I \models Q(\mathbf{a})$. The set of answers of $Q(\mathbf{x})$ w.r.t. \mathcal{F} and I is denoted as $\mathbf{cert}(Q, \mathcal{F}, I)$, where the free variables of $Q(\mathbf{x})$ are omitted for brevity.

3 Technical Approach

3.1 Overview

Given a TBox \mathcal{T} , an ABox \mathcal{A} and a query Q , our goal is to compute a (hopefully tight) upper bound to the set $\mathbf{cert}(Q, \mathcal{T}, \mathcal{A})$ of answers. We proceed as follows:

1. Transform \mathcal{T} into a set $\Sigma_{\mathcal{T}}$ of datalog ^{\pm, \vee} rules such that $\Sigma_{\mathcal{T}}$ is a conservative extension of \mathcal{T} .
2. Transform $\Sigma_{\mathcal{T}}$ into a set $\mathbf{approx}(\Sigma_{\mathcal{T}})$ of datalog rules s.t. $\mathbf{approx}(\Sigma_{\mathcal{T}}) \models \Sigma_{\mathcal{T}}$.

Student \sqsubseteq Person	\rightsquigarrow Student(x) \rightarrow Person(x)
RA \sqsubseteq Student	\rightsquigarrow RA(x) \rightarrow Student(x)
RA \sqsubseteq \exists works.Group	\rightsquigarrow RA(x) \rightarrow $\exists y$.[works(x, y) \wedge Group(y)]
Group \sqsubseteq Org	\rightsquigarrow Group(x) \rightarrow Org(x)
	Emp(x) \rightarrow Person(x)
Emp \equiv Person \sqcap \exists works.Org	\rightsquigarrow Emp(x) \rightarrow $\exists y$.[works(x, y) \wedge Org(y)]
	Person(x) \wedge works(x, y) \wedge Org(y) \rightarrow Emp(x)
works \sqsubseteq memberOf	\rightsquigarrow works(x, y) \rightarrow memberOf(x, y)
Student \sqsubseteq Grad \sqcup Undergrad	\rightsquigarrow Student(x) \rightarrow Grad(x) \vee Undergrad(x)
func(works)	\rightsquigarrow works(x, y) \wedge works(x, z) \rightarrow $y \approx z$

Fig. 1. Transforming \mathcal{T}_{ex} into datalog $^{\pm, \vee}$ rules $\Sigma_{\mathcal{T}_{\text{ex}}}$

3. Transform $\text{approx}(\Sigma_{\mathcal{T}})$ into an OWL 2 RL TBox \mathcal{T}' for which we have $\text{cert}(Q, \text{approx}(\Sigma_{\mathcal{T}}), \mathcal{A}) = \text{cert}(Q, \mathcal{T}', \mathcal{A})$ for every query Q and ABox \mathcal{A} .

Our transformation depends only on \mathcal{T} , and satisfies the following property:

$$\text{cert}(Q, \mathcal{T}, \mathcal{A}) \subseteq \text{cert}(Q, \mathcal{T}', \mathcal{A}) \quad \text{for every query } Q \text{ and ABox } \mathcal{A}$$

Given the OWL 2 RL TBox \mathcal{T}' we can then use \mathcal{T} and \mathcal{T}' with a materialisation based reasoner rl that is sound for OWL 2 and complete for OWL 2 RL (such as OWLIm) to respectively compute a lower and an upper bound to query answers for the given Q and \mathcal{A} . More precisely, we have:

$$\text{rl}(Q, \mathcal{T}, \mathcal{A}) \subseteq \text{cert}(Q, \mathcal{T}, \mathcal{A}) \subseteq \text{rl}(Q, \mathcal{T}', \mathcal{A}) \quad \text{for every } Q \text{ and } \mathcal{A}$$

3.2 Computing the Upper Bound

We next describe in detail the transformations in Steps 1-3. As a running example, we use the TBox \mathcal{T}_{ex} in Figure 1.

From DL TBoxes to Datalog $^{\pm, \vee}$ Rules. The first step is to transform the DL TBox \mathcal{T} into a set $\Sigma_{\mathcal{T}}$ of datalog $^{\pm, \vee}$ rules such that $\Sigma_{\mathcal{T}}$ is a conservative extension of \mathcal{T} . For a wide range of DLs, this can be achieved by first using the well-known correspondence between DL axioms and first-order logic formulas and then applying standard structural transformation rules, which may involve introducing new predicates. The transformation of our example \mathcal{T}_{ex} into datalog $^{\pm, \vee}$ rules $\Sigma_{\mathcal{T}_{\text{ex}}}$ is also shown in Figure 1; here, \mathcal{T}_{ex} and $\Sigma_{\mathcal{T}_{\text{ex}}}$ are equivalent.

From Datalog $^{\pm, \vee}$ to Datalog. Next, we approximate the datalog $^{\pm, \vee}$ rules $\Sigma_{\mathcal{T}}$ by a datalog program $\text{approx}(\Sigma_{\mathcal{T}})$ that entails $\Sigma_{\mathcal{T}}$. Intuitively, this transformation can be described in two steps:

- Rewrite each datalog $^{\pm, \vee}$ rule r into a set of datalog $^{\pm}$ rules by transforming disjunctions in the head of r into conjunctions and splitting the conjunctions into different datalog $^{\pm}$ rules. Such a way to eliminate disjunction in the head has been used in OWL reasoning with logic program [25].

$\begin{aligned} \text{RA}(x) \rightarrow \exists y. [\text{works}(x, y) \wedge \text{Group}(y)] &\rightsquigarrow \text{RA}(x) \rightarrow \text{works}(x, c_1) \wedge \text{Group}(c_1) \\ \text{Emp}(x) \rightarrow \exists y. [\text{works}(x, y) \wedge \text{Org}(y)] &\rightsquigarrow \text{Emp}(x) \rightarrow \text{works}(x, c_2) \wedge \text{Org}(c_2) \\ \text{Student}(x) \rightarrow \text{Grad}(x) \vee \text{Undergrad}(x) &\rightsquigarrow \text{Student}(x) \rightarrow \text{Grad}(x) \wedge \text{Undergrad}(x) \end{aligned}$

Fig. 2. Transforming $\Sigma_{\mathcal{T}_{\text{ex}}}$ into $\text{approx}(\Sigma_{\mathcal{T}_{\text{ex}}})$

- Transform the resulting datalog[±] rules into datalog using fresh individuals to skolemise existentially quantified variables.

Figure 2 illustrates this process for our example. The figure shows only the rules in $\Sigma_{\mathcal{T}_{\text{ex}}}$ that need changing; note that c_1 and c_2 are fresh individuals used to skolemise existentially quantified variables.¹

Definition 1. For each datalog^{±,∨} rule r of the form (1) and each variable $y_{ij} \in \mathbf{y}_i$, let c_{ij}^r be a fresh individual unique for r and y_{ij} . Then, $\text{approx}(r)$ is the following set of rules, where for each $1 \leq i \leq m$, θ_i^r is a substitution mapping each variable $y_{ij} \in \mathbf{y}_i$ to c_{ij}^r :

$$\text{approx}(r) = \{B_1 \wedge \dots \wedge B_n \rightarrow \varphi_i(\mathbf{x}, \theta_i^r(\mathbf{y}_i)) \mid 1 \leq i \leq m\} \quad (2)$$

For Σ a set of datalog^{±,∨} rules, $\text{approx}(\Sigma)$ is the smallest set that contains $\text{approx}(r)$ for each rule $r \in \Sigma$.

We next show that $\text{approx}(\Sigma_{\mathcal{T}})$ entails $\Sigma_{\mathcal{T}}$. The following proposition provides sufficient and necessary conditions for a datalog^{±,∨} rule to be entailed by an arbitrary set of first-order sentences (the proof is rather straightforward and can be found in [7]).

Proposition 1. Let \mathcal{F} be a set of first-order sentences and r be a datalog^{±,∨} rule of the form (1). Then, for each substitution σ mapping the free variables of r to distinct individuals not occurring in \mathcal{F} or r , we have $\mathcal{F} \models r$ if and only if

$$\mathcal{F} \cup \{\sigma(B_1), \dots, \sigma(B_n)\} \models \bigvee_{i=1}^m \exists \mathbf{y}_i. \varphi_i(\sigma(\mathbf{x}), \mathbf{y}_i)$$

Proposition 1 can be used to show that each datalog^{±,∨} rule r in $\Sigma_{\mathcal{T}}$ is entailed by $\text{approx}(r)$, and hence $\text{approx}(\Sigma_{\mathcal{T}})$ strengthens $\Sigma_{\mathcal{T}}$.

Proposition 2. For Σ a set of datalog^{±,∨} rules, we have $\text{approx}(\Sigma) \models \Sigma$.

Proof. It suffices to show that, for each rule $r \in \Sigma$ of the form (1) and each $r_i \in \text{approx}(r)$, we have $r_i \models r$. Let σ be a substitution mapping the free variables in r to fresh individuals; by Proposition 1, we have

$$r_i \models r \Leftrightarrow r_i \cup \{\sigma(B_1), \dots, \sigma(B_n)\} \models \bigvee_{i=1}^m \exists \mathbf{y}_i. \varphi_i(\sigma(\mathbf{x}), \mathbf{y}_i) \quad (3)$$

¹ Note that, although $\text{approx}(\Sigma_{\mathcal{T}_{\text{ex}}})$ is strictly speaking not datalog (we have conjunction of atoms in the head), it is trivially equivalent to a set of datalog rules.

Since r_i and r have the same body atoms, the definition of r_i in (2) implies

$$r_i \cup \{\sigma(B_1), \dots, \sigma(B_n)\} \models \varphi_i(\sigma(\mathbf{x}), \theta_i^r(\mathbf{y}_i)) \quad (4)$$

Since substitution θ_i^r maps variables to constants, the following conditions clearly hold by the semantics of first-order logic:

$$\varphi_i(\sigma(\mathbf{x}), \theta_i^r(\mathbf{y}_i)) \models \exists \mathbf{y}_i. \varphi_i(\sigma(\mathbf{x}), \mathbf{y}_i) \models \bigvee_{i=1}^m \exists \mathbf{y}_i. \varphi_i(\sigma(\mathbf{x}), \mathbf{y}_i) \quad (5)$$

But then, conditions (3), (4) and (5) immediately imply $r_i \models r$, as required. \square

The fact that $\mathbf{approx}(\Sigma)$ entails Σ implies that query answers w.r.t. $\mathbf{approx}(\Sigma)$ are an upper bound to those w.r.t. Σ .

Proposition 3. *For each set of datalog^{±,∨} rules Σ , each ABox \mathcal{A} and each query Q , we have $\mathbf{cert}(Q, \Sigma, \mathcal{A}) \subseteq \mathbf{cert}(Q, \mathbf{approx}(\Sigma), \mathcal{A})$.*

From Datalog to OWL 2 RL. The last step is to transform $\mathbf{approx}(\Sigma_{\mathcal{T}})$ into an OWL 2 RL TBox \mathcal{T}' . Although arbitrary datalog rules cannot be transformed into OWL 2 RL, the rules in $\mathbf{approx}(\Sigma_{\mathcal{T}})$ have been obtained from a DL TBox and are therefore tree-shaped, which makes this transformation possible.

There is, however, a technical consideration related to the fresh skolem constants in $\mathbf{approx}(\Sigma_{\mathcal{T}})$. In particular, the following rule in our running example (see Figure 2) does not directly correspond to an OWL 2 RL axiom:

$$\mathbf{RA}(x) \rightarrow \mathbf{works}(x, c_1) \wedge \mathbf{Group}(c_1) \quad (6)$$

This issue can be easily addressed by introducing fresh atomic roles. More precisely, rule (6) can be transformed into the following three OWL 2 RL axioms, where R' is a fresh atomic role:

$$\mathbf{RA} \sqsubseteq \exists R'. \{c_1\} \quad \top \sqsubseteq \forall R'. \mathbf{Group} \quad R' \sqsubseteq \mathbf{works}$$

3.3 Additional Considerations

Transformation of Disjunctive Rules. The proof of Proposition 2 suggests that we can easily weaken $\mathbf{approx}(\Sigma_{\mathcal{T}})$ given in Definition 1 such that $\Sigma_{\mathcal{T}}$ is still entailed. In particular, when transforming a disjunctive rule in $\Sigma_{\mathcal{T}}$ into datalog by replacing disjunctions with conjunctions, it suffices to keep only one of the conjuncts. For example, given the transformation

$$\mathbf{Student}(x) \rightarrow \mathbf{Grad}(x) \vee \mathbf{Undergrad}(x) \rightsquigarrow \mathbf{Student}(x) \rightarrow \mathbf{Grad}(x) \wedge \mathbf{Undergrad}(x)$$

we can replace in $\mathbf{approx}(\Sigma_{\mathcal{T}})$ the rule $\mathbf{Student}(x) \rightarrow \mathbf{Grad}(x) \wedge \mathbf{Undergrad}(x)$ with either $\mathbf{Student}(x) \rightarrow \mathbf{Grad}(x)$, or $\mathbf{Student}(x) \rightarrow \mathbf{Undergrad}(x)$. Any of these choices will lead to a (different) upper bound. In practice, one can choose to process any number of such different options, or simply devise suitable heuristics to choose the most promising one.

Unsatisfiability Issues. For given TBox \mathcal{T} and ABox \mathcal{A} , it might be the case that $\text{approx}(\Sigma_{\mathcal{T}}) \cup \mathcal{A}$ is unsatisfiable, whereas $\Sigma_{\mathcal{T}} \cup \mathcal{A}$ is satisfiable. Then, the upper bound we obtain is the trivial one for each query. For example, if we extend \mathcal{T}_{ex} in Figure 1 with the axiom $\text{Grad} \sqcap \text{Undergrad} \sqsubseteq \perp$, we obtain a rule with \perp in the head in $\Sigma_{\mathcal{T}}$, namely $\text{Grad}(x) \wedge \text{Undergrad}(x) \rightarrow \perp$. For $\mathcal{A}_{\text{ex}} = \{\text{RA}(a)\}$ we then have that $\mathcal{T}_{\text{ex}} \cup \mathcal{A}_{\text{ex}}$ is satisfiable, but $\text{approx}(\Sigma_{\mathcal{T}_{\text{ex}}}) \cup \mathcal{A}_{\text{ex}}$ is unsatisfiable; thus, for a given Q , any tuple of individuals of the appropriate arity must be included in the upper bound.

A way to address this issue is to remove from $\text{approx}(\Sigma_{\mathcal{T}})$ all rules with \perp in the head. Although it is then no longer the case that $\text{approx}(\Sigma_{\mathcal{T}}) \models \Sigma_{\mathcal{T}}$, we can still guarantee completeness provided that $\Sigma_{\mathcal{T}} \cup \mathcal{A}$ is satisfiable.

Proposition 4. *Let Σ be a set of datalog ^{\pm, \vee} rules and let Σ' the result of removing from $\text{approx}(\Sigma)$ all rules containing \perp in the head. Then, the following condition holds for each ABox \mathcal{A} and each query Q : if $\Sigma \cup \mathcal{A}$ is satisfiable, then $\text{cert}(Q, \Sigma, \mathcal{A}) \subseteq \text{cert}(Q, \Sigma', \mathcal{A})$.*

In practice, checking the satisfiability of $\mathcal{T} \cup \mathcal{A}$, which is equisatisfiable with $\Sigma_{\mathcal{T}} \cup \mathcal{A}$, is easier than (and a prerequisite for) query answering. Even if satisfiability of $\mathcal{T} \cup \mathcal{A}$ cannot be checked in practice using a complete reasoner for a very large \mathcal{A} , we can still compute an upper bound “modulo satisfiability”.

Why Translating Back into OWL 2 RL? Our last step was to transform $\text{approx}(\Sigma_{\mathcal{T}})$ into OWL 2 RL TBox \mathcal{T}' . Note, however, that one could obtain the upper bound directly from $\text{approx}(\Sigma_{\mathcal{T}})$ by first computing the saturation \mathcal{A}' of $\text{approx}(\Sigma_{\mathcal{T}})$ w.r.t. \mathcal{A} and then computing $\text{cert}(Q, \emptyset, \mathcal{A}')$.

The saturation \mathcal{A}' can be computed in polynomial time [5]; indeed, the rules in $\text{approx}(\Sigma_{\mathcal{T}})$ are tree-shaped, which can be exploited to obtain a polynomial time saturation procedure. This could lead to better performance in the computation of our upper bound —an interesting topic for future work.

Our current approach, however, does have some advantages. In particular, one can use the same off-the-shelf RL reasoner (such as OWLim) to compute both the lower and upper bound, which is convenient in practice. Furthermore, as suggested by our experiments, reasoners such as OWLim are quite efficient for large data sets.

4 Experiments

We have implemented our approach in Java and used the latest version of OWLim-Lite as an OWL 2 RL reasoner. All experiments have been performed on a desktop computer with 4Gb of RAM, of which 3000Mb were assigned as maximum heap size in Java.

In our experiments, we have used LUBM extended with additional synthetically generated queries. The LUBM ontology contains 93 TBox axioms, out of which 8 contain existential quantification, and 39 are domain and range axioms. The LUBM ontology, however, does not contain disjunction or negation; thus,

Table 1. Number of answers for the 14 LUBM queries

Query	1	2	3	4	5	6	7
Lower Bound	4	0	6	34	719	7356	67
Upper Bound	4	0	6	34	719	7356	67
Query	8	9	10	11	12	13	14
Lower Bound	7356	194	4	212	14	1	5594
Upper Bound	7356	194	4	212	14	1	5594

Table 2. Synthetic queries for which OWLim is incomplete

Query	Lower Bound	Upper Bound	HermiT’s answers
3	540	1087	1087
51	0	547	547
67	540	1087	1087
69	0	547	547

the corresponding issues discussed in Section 3.3 do not apply to our experiments. To test how tight our upper bounds are for different queries, we have used LUBM(1,0), the smallest data set in the benchmark, since the complete DL reasoner HermiT can answer all test queries for this data set. LUBM(1,0) contains data for one university and 14 departments, with a total of 100,543 ABox assertions about 17,174 different individuals.

4.1 Results for LUBM(1,0)

Standard Queries. LUBM provides 14 queries. As shown in Table 1, the lower and upper bounds computed using OWLim coincide, which allows us to conclude that OWLim is complete for all these queries and the LUBM(1,0) data set. Hence, in these cases, one wouldn’t need to use a complete DL reasoner at all.

Additional Synthetic Queries. We have used a synthetic query generation tool [9] to compute 78 additional queries for LUBM. For all these queries, except those in Table 2, lower and upper bounds also coincide. Furthermore, for all queries in Table 2 the upper bound is tight.

Observe, however, that the lower bound is missing those answers which require taking into account LUBM’s axioms with existential quantifiers, for which OWLim is not complete. For example, consider the following query

$$Q_{51}(x) = \exists y.(\text{memberOf}(x, y) \wedge \text{Group}(y))$$

LUBM’s ontology contains all the axioms in \mathcal{T}_{ex} , except for the last two axioms in Figure 1; furthermore, LUBM(1,0) contains many instances of RA. Since LUBM’s ontology implies that each RA works for (and hence is a member of) some group, all these instances are answers to Q_{51} , which are not computed by OWLim.

Additional Query. For all previous test queries, we have obtained a tight upper bound. To show that this is not always the case, we have manually created the additional query given next.

$$Q(x_1, x_2) = \exists y. \text{works}(x_1, y) \wedge \text{works}(x_2, y) \wedge \text{Group}(y)$$

Since this query is not tree-like, it cannot be answered using HerMiT. To obtain the complete answers, we have used REQUIEM² to compute a (complete) UCQ rewriting U of Q w.r.t. LUBM’s ontology; then, we used OWLim to compute the answers to U w.r.t. the LUBM(1,0) data. For this query, the lower and upper bounds contained zero and 299,209 answers, respectively; in contrast, we obtained 547 answers using REQUIEM and hence the upper bound is not tight.

As shown in Figure 2, our transformation implies that all RAs work for the same group (represented by the skolem constant c_1); since there are many instances of RA in LUBM(1,0), all pairs of RA instances will be included in the upper bound. Clearly, however, many of these RAs work for different groups.

Note, however, that even for this query we managed to reduce the number of candidate answers from $17,174^2 \sim 3 \times 10^8$ to 299,209.

4.2 Scalability Tests

To test scalability of upper bound computation, we have performed additional experiments using LUBM data sets of increasing size (from 1 to 10 universities). For each data set and each of the 78 synthetic queries, we have computed lower and upper bounds (using OWLim) and the complete answers (using HerMiT). The results are summarised in Figure 4.2, where the time refers to the total query answering time for all test queries.

We can observe similar query answering times and scalability behaviour for lower and upper bound computation. Furthermore, we can observe that HerMiT’s performance is similar to OWLim’s for relatively small data sets. HerMiT, however, does not scale well for the largest LUBM data sets. In particular, it takes 178s to complete the tests for 9 universities and runs out of memory for 10 universities; in contrast, OWLim computation times increase more regularly.

5 Conclusion and Future Work

In this paper, we have proposed a novel technique for efficiently computing an upper bound to CQ answers for ontologies given in a expressive DL. Our preliminary experiments on LUBM show that this upper bound might be tight in many cases. Furthermore, we identified cases where lower and upper bounds coincide and hence it is not necessary to use a fully-fledged DL reasoner such as HerMiT to compute query answers (HerMiT is able to answer rollable queries).

Our work so far, however, is only very preliminary, and there are plenty of possibilities for future work. We are planning to perform experiments with ontologies involving disjunction and negation, and address the issues discussed in

² <http://www.cs.ox.ac.uk/isg/tools/Requiem/>

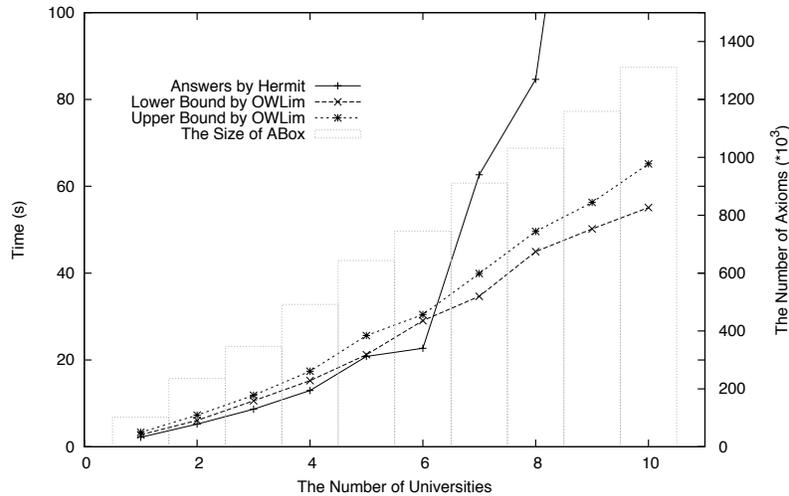


Fig. 3. Running Time Comparison

Section 3.3. Furthermore, we will implement a dedicated datalog engine that can compute the saturation in polynomial time for tree-shaped datalog rules; this might allow us to improve performance as well as to simultaneously compute lower and upper bounds. We will also develop techniques for identifying during upper bound computation the fragments of the ABox and TBox that are sufficient to determine, using a complete reasoner, which of the answers in the upper bound are actual answers; we expect that these techniques will provide additional room for optimisation. As a result, we can integrate all these techniques in HermiT to optimise query answering.

Acknowledgements. This work was supported by the Royal Society, the EU FP7 project SEALS and the EPSRC projects ConDOR, ExODA, and LogMap.

References

1. Baader, F., Sertkaya, B., Turhan, A.: Computing the least common subsumer wrt a background terminology. *J. of Applied Logic (JAL)* 5(3), 392–420 (2007)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: *IJCAI* (2005)
3. Cali, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: *Proc. of LICS* (2010)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning (JAR)* 39(3), 385–429 (2007)

5. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity conditions and their application to query answering in description logics. In: Proc. of KR (2012)
6. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Semantics (JWS) 6(4), 309–322 (2008)
7. Cuenca Grau, B., Motik, B., Stoilos, G., Horrocks, I.: Completeness guarantees for incomplete ontology reasoners: Theory and practice. J. of Artificial Intelligence Research (JAIR) 43 (2012)
8. Del Val, A.: First order lub approximations: characterization and algorithms. Artificial Intelligence 162(1-2), 7–48 (2005)
9. Grau, B.C., Stoilos, G.: What to ask to an incomplete semantic web reasoner? In: Proc. of IJCAI. pp. 419–476 (2011)
10. Grosz, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proc. of WWW (2003)
11. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for OWL knowledge base systems. J. Web Semantics (JWS) 3(2-3), 158–182 (2005)
12. Haarslev, V., Möller, R.: RACER system description. J. of Automated Reasoning (JAR) pp. 701–705 (2001)
13. Haarslev, V., Möller, R., Wessel, M.: Querying the semantic web with RACER+NRQL. In: Proc. of ADL (2004)
14. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SROIQ*. In: Proc. of KR (2006)
15. Kazakov, Y.: *RIQ* and *SROIQ* are harder than *SHOIQ*. In: Proc. of KR (2008)
16. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM—a pragmatic semantic repository for OWL. In: Proc. of WISE Workshops (2005)
17. Kollia, I., Glimm, B., Horrocks, I.: Query answering over sroi q knowledge bases with SPARQL. In: Proc. of DL (2011)
18. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic \mathcal{EL} using a relational database system. In: Proc. of IJCAI. vol. 9 (2009)
19. McBride, B.: Jena: Implementing the rdf model and syntax specification. In: Semantic Web Workshop. vol. 40 (2001)
20. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research (JAIR) 36(1), 165–228 (2009)
21. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation (2009)
22. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. of Applied Logic (JAL) 8(2), 186–209 (2010)
23. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for tbox reasoning. In: Proc. of AAAI (2010)
24. Rosati, R.: On conjunctive query answering in \mathcal{EL} . In: Proc. of DL. vol. 46 (2007)
25. Rudolph, S., Krötzsch, M., Hitzler, P., Sintek, M., Vrandečić, D.: Efficient owl reasoning with logic programs—evaluations. Web Reasoning and Rule Systems (2007)
26. Rudolph, S., Glimm, B.: Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend! J. Artif. Intell. Res. (JAIR) 39, 429–481 (2010)
27. Selman, B., Kautz, H.: Knowledge compilation and theory approximation. J. of the ACM (JACM) 43(2), 193–224 (1996)
28. Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: System description. J. Automated Reasoning (JAR) pp. 292–297 (2006)

Author Index

A

Abiteboul, Serge	1
Alechina, Natasha	290
Arenas, Marcelo	4
Armas Romero, Ana	15
Artale, Alessandro	312
Aslani, Mina	400

B

Baader, Franz	26, 37, 585
Bail, Samantha	48
Bellodi, Elena	519
Bienvenu, Meghyn	59, 70, 81
Bonatti, Piero	2
Borgida, Alexander	92
Borgwardt, Stefan	26, 37, 103, 411
Botoeva, Elena	4
Bozzato, Loris	114
Britz, Katarina	443

C

Calvanese, Diego	4, 312
Colucci, Simona	125
Cuenca Grau, Bernardo	15, 596
Cure, Olivier	136

D

Distel, Felix	103
Donini, Francesco M.	125
Le Duc, Chan	136

E

Ecke, Andreas	147
Eiter, Thomas	158

F

Franconi, Enrico	169, 422
Fu, Weili	432

G

Giordano, Laura	180
Glimm, Birte	246, 552
Gliozzi, Valentina	180
Gonçalves, Rafael S.	191
Gottlob, Georg	202
Graves, Henson	497

H

Haarslev, Volker	378, 400, 530
------------------	---------------

Halland, Ken	443
Hernich, André	202
Homola, Martin	114
Horkoff, Jennifer	92
Horrocks, Ian	15, 279, 541, 596
Hudek, Alexander	367
I	
Ibanez-Garcia, Yazmin Angelica	213
K	
Karam, Naouel	454
Kerhet, Volha	169
Khodadadi, Mohammad	224
Kikot, Stanislav	235
Kollia, Ilianna	246
Kontchakov, Roman	235
Krdzavac, Nenad	497
Kupke, Clemens	202
L	
Lamma, Evelina	519
Lamolle, Myriam	136
Lee, Kevin	508
Lembo, Domenico	257
Lenzerini, Maurizio	257
Liebig, Thorsten	552
Lippmann, Marcel	585
Lisi, Francesca A.	464
Liu, Hongkai	585
Logan, Brian	290
Lukasiewicz, Thomas	202
Lutz, Carsten	70, 268
M	
Magka, Despoina	279
Malenko, Jaromir	475
Mendez, Julian	26
Meyer, Thomas	486
Moodley, Kody	486
Morawska, Barbara	26, 37
Mosca, Alessandro	422
Mosurovic, Milenko	497
Motik, Boris	279, 541
Mylopoulos, John	92
Möller, Ralf	301
N	
Ngo, Nhung	169
Nguyen, Hai H.	290

O	
Olivetti, Nicola	180
Ortiz, Magdalena	81, 158
Özcep, Özgür Lütfü	301
P	
Pan, Jeff Z.	508
Parsia, Bijan	48, 191, 574
Paschke, Adrian	454
Peñaloza, Rafael	103, 411, 432
Podolskii, Vladimir	235
Pozzato, Gian Luca	180
Q	
Queralt, Anna	312
R	
Rector, Alan	3
Ren, Yuan	508
Riguzzi, Fabrizio	519
Roosta Pour, Laleh	530
Rosati, Riccardo	92, 257, 323
Ruzzi, Marco	257
Ryzhikov, Vladislav	4
S	
Sattler, Ulrike	48, 191, 574
Savo, Domenico Fabio	257
Schmidt, Renate A.	224
Serafini, Luciano	114
Seylan, Inanc	268
Sherkhonov, Evgeny	4
Simancik, Frantisek	334
Simkus, Mantas	81, 158
Solomakhin, Dmitry	422
Soutchanski, Mikhail	389, 585
Stamou, Giorgos	356
Stefanoni, Giorgio	541
Steigmiller, Andreas	552
Stepanek, Petr	475
Stoilos, Giorgos	356
T	
Teniente, Ernest	312
Thomazo, Michaël	563
Tishkovsky, Dmitry	224
Toman, David	367
Tran, Trung-Kien	158
Tsarkov, Dmitry	345
Turhan, Anni-Yasmin	147

V	
Varzinczak, Ivan	486
Venetis, Tassos	356
Del Vescovo, Chiara	574
W	
Weddell, Grant	367
Wolter, Frank	70, 268
Wu, Jiewen	367
Wu, Kejia	378
X	
Xiao, Guohui	158
Y	
Yehia, Wael	389, 585
Z	
Zakharyashev, Michael	235
Zhou, Yujiao	596