Proceedings of the 7th International

Workshop on Modular Ontologies (WoMO) 2013

held at Corunna, Spain, on September 15, 2013
as a satellite event of LPNMR 2013

Workshop chairs and proceedings editors:

Chiara Del Vescovo, University of Manchester, UK
Torsten Hahmann, University of Toronto, Canada
David Pearce, Universidad Politécnica de Madrid, Spain
Dirk Walther, TU Dresden, Germany

Preface

Modularity has been and continues to be one of the central research topics in ontology engineering. The number of ontologies available, as well as their size, is steadily increasing. There is a large variation in subject matter, level of specification and detail, intended purpose and application. Ontologies covering different domains are often developed in a distributed manner; contributions from different sources cover different parts of a single domain. Not only is it difficult to determine and define interrelations between such distributed ontologies, it is also challenging to reconcile ontologies which might be consistent on their own but jointly inconsistent. Further challenges include extracting the relevant parts of an ontology, re-combining independently developed ontologies in order to form new ones, determining the modular structure of an ontology for comprehension, and the use of ontology modules to facilitate incremental reasoning and version control.

Modularity is envisaged to allow mechanisms for easy and flexible reuse, combination, generalization, structuring, maintenance, collaboration, design patterns, and comprehension. This is analogous to the role of modularity in software engineering, where there are well-understood notions of modularity that have led to generally accepted and widely supported mechanisms for the named tasks. In contrast, modularity for ontologies is still an active research field with open questions because existing approaches are heterogeneous and less universally applicable. For ontology engineering, modularity is central not only to reducing the complexity of understanding ontologies, but also to maintaining, querying and reasoning over modules. Distinctions between modules can be drawn on the basis of structural, semantic, or functional aspects, which can also be applied to compositions of ontologies or to indicate links between ontologies.

In particular, reuse and sharing of information and resources across ontologies depend on purpose-specific, logically versatile criteria. Such purposes include "tight" logical integration of different ontologies (wholly or in part), "loose" association and information exchange, the detection of overlapping parts, traversing through different ontologies, alignment of vocabularies, module extraction possibly respecting privacy concerns and hiding of information, etc. Another important aspect of modularity in ontologies is the problem of evaluating the *quality* of single modules or of the achieved overall modularization of an ontology. Again, such evaluations can be based on various (semantic or syntactic) criteria and employ a variety of statistical/heuristic or logical methods.

Recent research on ontology modularity has produced substantial results and approaches towards foundations of modularity, techniques of modularization and modular developments, distributed and incremental reasoning, as well as the use of modules in different application scenarios, providing a foundation for further research and development. Since the beginning of the WoMO workshop series, there has been growing interest in the modularization of ontologies, modular development of ontologies, and information exchange across different modular ontologies. In real life, however, integration problems are still mostly tackled

in an ad-hoc manner, with no clear notion of what to expect from the resulting ontological structure. Those methods are not always efficient, and they often lead to unintended consequences, even if the individual ontologies to be integrated are widely tested and understood.

Topics covered by WoMO include, but are not limited to:

What is Modularity?
- Kinds of modules and their properties
- Modules vs. contexts
- Design patterns
- Granularity of representation

Logical/Foundational Studies
- Conservativity and syntactic approximations for modules
- Modular ontology languages
- Reconciling inconsistencies across modules
- Formal structuring of modules
- Heterogeneity

Algorithmic Approaches
- Distributed and incremental reasoning
- Modularization and module extraction
- Sharing, linking, and reuse
- Hiding and privacy
- Evaluation of modularization approaches
- Complexity of reasoning
- Implemented systems

Application Areas
- Modularity in the Semantic Web
- Life Sciences
- Bio-Ontologies
- Natural Language Processing
- Ontologies of space and time
- Ambient intelligence
- Social intelligence
- Collaborative ontology development and ontology versioning

Previous events. The WoMO 2013 workshop follows a series of successful events that have been an excellent venue for practitioners and researchers to discuss latest work and current problems. It is intended to consolidate cutting-edge approaches that tackle the problem of ontological modularity and bring together researchers from different disciplines who study the problem of modularity in ontologies at a fundamental level, develop design tools for distributed ontology engineering, and apply modularity in different use cases and application scenarios. Previous editions of WoMO are listed below. The links refer to their homepages and proceedings.

WoMO 2006. The 1st workshop on modular ontologies, co-located with ISWC 2006, Athens, Georgia, USA. Invited speakers were Alex Borgida (Rutgers) and Frank Wolter (Liverpool). Organizers and program chairs were

http://www.cild.iastate.edu/events/womo.html
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-232

WoMO 2007. The 2nd workshop, co-located with K-CAP 2007, Whistler BC, Canada. The invited speaker was Ken Barker (Texas at Austin).

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-315

WoRM 2008. The 3rd workshop in the series, co-located with ESWC 2008, Tenerife, Spain, entitled 'Ontologies: Reasoning and Modularity' had a special emphasis on reasoning methods.

http://dkm.fbk.eu/worm08
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-348

WoMO 2010. The 4th workshop in the series, co-located with FOIS 2010, Toronto, Canada. Invited speakers were Simon Colton (London) and Marco Schorlemmer (Barcelona).

http://www.informatik.uni-bremen.de/%7Eokutz/womo4
http://www.booksonline.iospress.nl/Content/View.aspx?piid=16268

WoMO 2011. The 5th workshop in the series, co-located with ESSLLI 2011, Ljubljana, Slovenia. Invited speakers were Stefano Borgo (Trento), Stefan Schulz (Graz) and Michael Zakharyaschev (London).

http://www.informatik.uni-bremen.de/%7Eokutz/womo5
http://www.booksonline.iospress.nl/Content/View.aspx?piid=20369

WoMO 2012. The 6th workshop in the series, co-located with ICBO/FOIS 2012, Graz, Austria. Invited speakers were Thomas Eiter (Vienna) and Luciano Serafini (Trento).

http://www.informatik.uni-bremen.de/~ts/womo2012/
http://ceur-ws.org/Vol-875/

Organizers of the previous editions and editors of the proceedings were Diego Calvanese (Bozen-Bolzano) – 2008; Bernardo Cuenca Grau (Manchester, Oxford) – 2007, 2008, 2010; Peter Haase (Karlsruhe) – 2006; Jie Bao (Rensselaer) – 2010; Joana Hois (Bremen) – 2010; Vasant Honovar (Iowa State) – 2006, 2007; Oliver Kutz (Manchester, Bremen) – 2006, 2010, 2011; Ulrike Sattler (Manchester) – 2008; Anne Schlicht (Mannheim) – 2007; Thomas Schneider (Bremen) – 2011, 2012; Luciano Serafini (Trento) – 2008; Evren Sirin, (Clark & Parsia LLC, Washington DC) – 2008; York Sure (Karlsruhe) – 2006; Andrei Tamilin (Trento) – 2006, 2008; Michael Wessel (Hamburg) – 2008; Dirk Walther (Madrid) – 2012; Frank Wolter (Liverpool) – 2007, 2008

This volume contains the papers presented at the 7th International Workshop on Modular Ontologies (WoMO 2013) held on September 15, 2013 in Corunna,

Spain as a satellite event of the conference LPNMR 2013. We received 9 submissions. Each submission was reviewed by three program committee members. The committee decided to accept five papers for long or short presentations. The program also included two invited talks:

- Till Mossakowski (University of Bremen, Germany)
  The Distributed Ontology, Modeling and Specification Language
- George Vouros (University of Piraeus, Greece)
  Combining ontologies in settings with multiple agents

Acknowledgments. We would like to thank the PC members and the additional reviewers for their timely reviewing work, our invited speakers for delivering keynote presentations at the workshop, and the authors and participants for contributing to the workshop program. We would also like to thank the organizers of LPNMR 2013 for hosting the WoMO workshop, the IAOA and SINTELNET for their generous financial support, and the EasyChair developers for greatly simplifying the work of the program committee.

September 27, 2013
Manchester, Toronto, Madrid
and Dresden

Chiara Del Vescovo
Torsten Hahmann
David Pearce
Dirk Walther

Table of Contents

Summaries of Invited Talks

Regular Papers

Short Papers

Program Committee

Kenneth Baclawski	Northeastern University, Boston, USA
Eva Blomqvist	Linköping University, Sweden
Alex Borgida	Rutgers University, Piscataway, USA
Stefano Borgo	Laboratory for Applied Ontology, ISTC-CNR, Trento, Italy
Gerhard Brewka	University of Leipzig, Germany
Mike Dean	Raytheon BBN Technologies, Ann Arbor, USA
Chiara Del Vescovo (co-chair)	The University of Manchester, UK
Thomas Eiter	Technical University of Vienna, Austria
Pawel Garbacz	The John Paul II Catholic University of Lublin, Poland
Dagmar Gromann	Vienna University of Economics and Business, Austria
Michael Gruninger	University of Toronto, Canada
Torsten Hahmann (co-chair)	University of Toronto, Canada
Robert Hoehndorf	University of Cambridge, UK
Dieter Hutter	DFKI GmbH, Bremen, Germany
Tomi Janhunen	Aalto University, Finland
Pavel Klinov	University of Ulm, Germany
Christoph Lange	University of Birmingham, UK
Thomas Meyer	CSIR Meraka Institute, Pretoria, South Afrika
Leo Obrst	MITRE, McLean, VA, USA
David Pearce (co-chair)	Universidad Politécnica de Madrid, Spain
Marco Schorlemmer	IIIA-CSIC, Barcelona, Spain
Luciano Serafini	Fundazione Bruno Kessler, Trento, Italy
Dmitry Tsarkov	The University of Manchester, UK
Dirk Walther (co-chair)	TU Dresden, Germany

Author Index

The Distributed Ontology, Modelling and Specification Language – DOL

Till Mossakowski[1,2], Oliver Kutz[1], Mihai Codescu[3], and Christoph Lange[4]

[1] Collaborative Research Centre on Spatial Cognition, University of Bremen
[2] DFKI GmbH Bremen
[3] University of Erlangen-Nürnberg
[4] School of Computer Science, University of Birmingham

Abstract. There is a diversity of ontology languages in use, among them OWL, RDF, OBO, Common Logic, and F-logic. Related languages such as UML class diagrams, entity-relationship diagrams and object role modelling provide bridges from ontology modelling to applications, e.g. in software engineering and databases.
Another diversity appears at the level of ontology modularity and relations among ontologies. There is ontology matching and alignment, module extraction, interpolation, ontologies linked by bridges, interpretation and refinement, and combination of ontologies.
The Distributed Ontology, Modelling and Specification Language (DOL) aims at providing a unified meta language for handling this diversity. In particular, DOL provides constructs for (1) "as-is" use of ontologies formulated in a specific ontology language, (2) ontologies formalised in heterogeneous logics, (3) modular ontologies, and (4) links between ontologies. This paper sketches the design of the DOL language. DOL will be submitted as a proposal within the OntoIOp (Ontology Integration and Interoperability) standardisation activity of the Object Management Group (OMG).

1 Introduction

OWL is a popular language for ontologies.[5] Yet, the restriction to a decidable description logic often hinders ontology designers from expressing knowledge that cannot (or can only in quite complicated ways) be expressed in a description logic. A practice to deal with this problem is to intersperse OWL ontologies with first-order axioms, e.g. in the case of bio-ontologies where mereological relations such as parthood are of great importance, though only partly definable in OWL. However, these remain informal annotations to inform the human designer, rather than first-class citizens of the ontology with formal semantics and impact on reasoning. One goal of the Distributed Ontology, Modelling and Specification Language (DOL), discussed in detail in this paper, is therefore to equip such heterogeneous ontologies with a precise semantics and proof theory.

[5] We adopt the completely formal position that an ontology is a formal theory in a given ontology language, and that an ontology language is any logical language that some community considers suitable for ontology design.

A variety of languages is used for formalising ontologies. Some of these, such as RDF (mostly used for linked data), OBO and certain[6] UML class diagrams, can be seen more or less as fragments and notational variants of OWL, while others, such as F-logic and Common Logic (CL), clearly go beyond the expressiveness of OWL.

We face this diversity not by proposing yet another ontology language that would subsume all the others, but by accepting this pluralism in ontology languages and by formulating means (on a sound and formal semantic basis) to compare and integrate ontologies written in different formalisms. This view is a bit different from that of unifying languages such as OWL and CL, which are meant to be "universal" formalisms (for a certain domain/application field), into which everything else can be mapped and represented. While such "universal" formalisms are clearly important and helpful for reducing the diversity of formalisms, it is still a matter of fact that no single formalism will be the Esperanto that is used by everybody [23]. It is therefore important to both accept the existing diversity of formalisms and to provide means of organising their coexistence in a way that enables formal interoperability among ontologies.

DOL enjoys the following distinctive features:

- modular and distributed ontologies are specially supported,
- ontologies can not only be aligned (as in BioPortal [37] and NeON [14]), but also combined along alignments,
- logical links between ontologies (interpretation of theories, conservative extensions etc.) are supported,
- support for a variety of ontology languages (OWL, RDF, Common Logic, first-order logic; planned: UML, relational database schemas, F-logic, distributed description logics, and more),
- ontologies can be translated to other ontology languages, and compared with ontologies in other languages,
- heterogeneous ontologies involving several languages can be built,
- ontology languages and ontology language translations are first-class citizens and are available on the Web as linked data.

The paper is organised as follows: we first discuss the theoretical foundations of DOL in Section 2, followed by a sketch of the DOL language itself in Section 3. Section 4 briefly discusses the DOL-enabled, web-based ontology repository engine Ontohub, and Section 5 concludes.

2 Foundations of the Distributed Ontology, Modelling and Specification Language (DOL)

The Distributed Ontology, Modelling and Specification Language (DOL)[7] aims at providing a unified framework for (1) "as-is" use of ontologies formulated in

[6] Those avoiding qualified associations (amounting to identification constraints), n-ary relations (for $n > 2$) and stereotyping.

[7] DOL has formerly been standardised within ISO/TC 37/SC 3. The OntoIOp (Ontology Integration and Interoperability) activity is now being continued at OMG, see the project page at `http://ontoiop.org`.

a specific ontology language, (2) ontologies formalised in heterogeneous logics, (3) modular ontologies, and (4) links between ontologies. Historically, the design of DOL has inherited many ideas and features (1) discussed in the Workshop on Modular Ontologies series [13, 12, 39, 19, 24, 40], (2) from the Alignment API [9], and (3) from the CASL (Common Algebraic Specification Language) and HetCASL (CASL's heterogeneous extension) languages, standardised in IFIP WG 1.3[8] (Foundations of System Specification) [2, 27, 32, 20].

A distributed ontology in DOL consists of modules formalised in *basic ontology languages*, such as OWL (based on description logic) or Common Logic (based on first-order logic with some second-order features). These modules are serialised in the existing syntaxes of these languages in order to facilitate reuse of existing ontologies. DOL adds a meta-level on top, which allows for expressing heterogeneous ontologies and links between ontologies.[9] Such links include (heterogeneous) *imports* and *alignments*, *conservative extensions* (important for studying ontology modules), and *theory interpretations* (important for reusing proofs). Thus, DOL gives ontology interoperability a formal grounding and makes heterogeneous ontologies and services based on them amenable to automated verification. The basic syntax and semantics of DOL has been introduced in [35, 34], and the general theory of heterogeneous specifications for ontologies in [22]. DOL uses internationalised resource identifiers (IRIs, the Unicode-aware superset of URIs) for all entities of distributed ontologies to make them referenceable on the Web.

2.1 Foundations

The large variety of logical languages in use can be captured at an abstract level using the concept of *institutions* [10]. This allows us to develop results independently of the particularities of a logical system and to use the notions of institution and logical language interchangeably throughout the rest of the paper. The main idea is to collect the non-logical symbols of the language in signatures and to assign to each signature the set of sentences that can be formed with its symbols. For each signature, we provide means for extracting the symbols it consists of, together with their kind. Signature morphisms are mappings between signatures. We do not assume any details except that signature morphisms can be composed and that there are identity morphisms; this amounts to a category of signatures. Readers unfamiliar with category theory may replace this with a partial order (signature morphisms are then just inclusions). See [34] for details of this simplified foundation.

Institutions also provide a model theory, which introduces semantics for the language and gives a satisfaction relation between the models and the sentences of a signature. The only restriction imposed is the satisfaction condition, which captures the idea that truth is invariant under change of notation (and enlargement of context) along signature morphisms. This relies on two further components of institutions: the translation of sentences along signature morphisms, and

[8] See `http://ifipwg13.informatik.uni-bremen.de`

[9] The languages that we call "basic" ontology languages here are usually limited to one logic and do not provide meta-theoretical constructs.

the reduction of models against signature morphisms (generalising the notion of model reduct known from logic).

It is also possible to complement an institution with a proof theory, introducing a derivability relation between sentences, formalised as an *entailment system* [30]. In particular, this can be done for all logics that have so far been in use in DOL.

Example 1. OWL signatures consist of sets of atomic classes, individuals and properties. OWL signature morphisms map classes to classes, individuals to individuals, and properties to properties. For an OWL signature Σ, sentences are subsumption relations between classes or properties, membership assertions of individuals in classes and pairs of individuals in properties, complex role inclusions, and some more. Sentence translation along a signature morphism simply replaces non-logical symbols with their image along the morphism. The kinds of symbols are class, individual, object property and data property, respectively, and the set of symbols of a signature is the union of its sets of classes, individuals and properties. Models are (unsorted) first-order structures that interpret concepts as unary and properties as binary predicates, and individuals as elements of the universe of the structure, and satisfaction is the standard satisfaction of description logics. This gives us an institution for OWL.

In this framework, a basic ontology O over an institution I is a pair (Σ, E) where Σ is a signature and E is a set of Σ-sentences. Given a basic ontology O, we denote by $\mathsf{Sig}(O)$ the signature of the ontology. An ontology morphism $\sigma : (\Sigma_1, E_1) \to (\Sigma_2, E_2)$ is a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ such that $\sigma(E_1)$ is a logical consequence of E_2.

Several notions of *translations* between institutions can be introduced. The most frequently used variant are *institution comorphisms* [11]. A comorphism from institution L_1 to institution L_2 maps L_1-signatures to L_2-signatures along a functor Φ and Σ-sentences in L_1 to $\Phi(\Sigma)$-sentences in L_2, for each L_1-signature Σ, while $\Phi(\Sigma)$-models are mapped to Σ-models. Again, a satisfaction condition has to be fulfilled. For *institution morphisms*, the directions of the translation of sentences and models are reversed. See [11] for full details.

Figure 1 shows a conceptual hierarchy of mappings.[10] Mappings are split along the following dichotomies:

- *translation* versus *projection*: a translation embeds or encodes a logic into another one, while a projection is a forgetful operation (e.g. the projection from first-order logic to propositional logic forgets predicates with arity greater than zero). Technically, the distinction is that between institution comorphisms and morphisms.
- *plain mapping* versus *simple theoroidal mapping* [11]: while a plain mapping needs to map signatures to signatures, a simple theoroidal mapping maps signatures to theories. The latter therefore allows for using "infrastructure axioms": e.g. when mapping OWL to Common Logic, it is convenient to rely on a first-order axiomatisation of a transitivity predicate for properties.

[10] This graph, computed within PROTÉGÉ, shows the inferred class hierarchy below the class `Mapping` of the LoLa ontology (see Section 2.3 below).

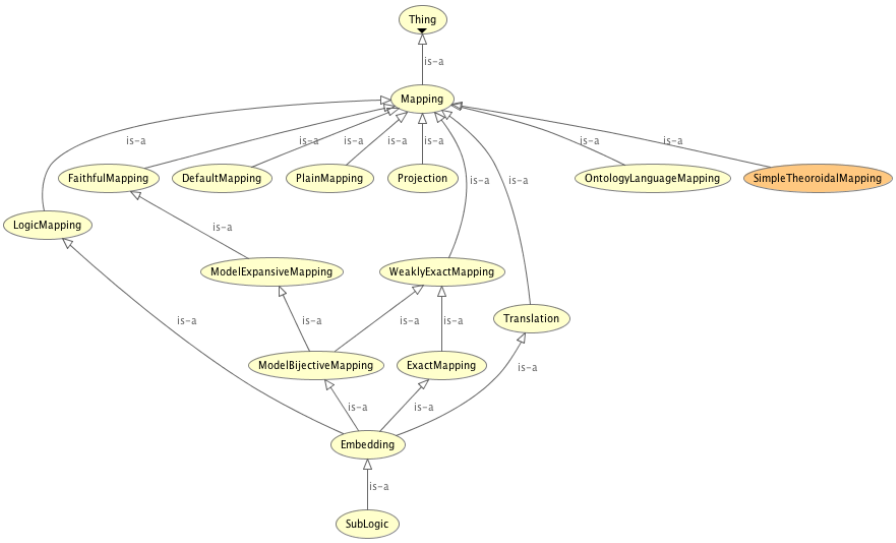


Fig. 1. The mapping subset of the LoLa ontology

Mappings can also be classified according to their accuracy; see [33] for details. *Sublogics* are the most accurate mappings: they are syntactic subsets. *Embeddings* come close to sublogics, like injective functions come close to subsets. A mapping can be *faithful* in the sense that logical consequence (or logical deduction) is preserved and reflected, that is, inference systems and reasoning engines for the target logic can be reused for the source logic (along the mapping). *(Weak) exactness* is a technical property that guarantees this faithfulness even in the presences of ontology structuring operations [5].

2.2 A Graph of Logic Translations

Figure 2 is a revised and extended version of the graph of logics and translations introduced in [33]. New nodes include UML class diagrams, OWL-Full (i.e. OWL with an RDF semantics instead of description logic semantics), and Common Logic without second-order features (CL^-). We have defined the translations between most of these logics in earlier publications [35, 33]. The definitions of the DOL conformance of some central standard ontology languages and translations among them will be given as annexes to the standard and published in an open registry, which is also the place where the remaining definitions will be maintained (cf. Section 2.3).

2.3 A Registry for Ontology Languages and Mappings

Beyond those shown so far, it will be possible to use any (future) language or mapping (in the sense of Section 2.1) with DOL. We host a *registry* to which

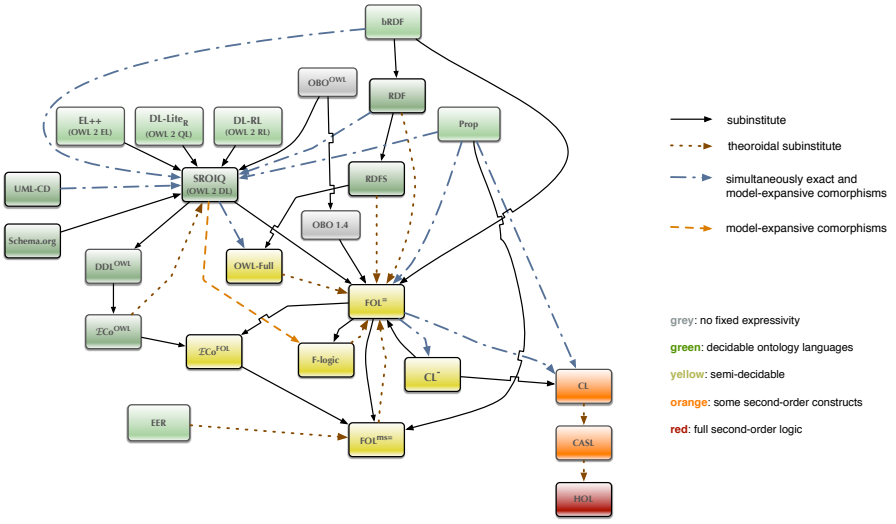


Fig. 2. The current logic translation graph for DOL-conforming languages

the community can contribute descriptions of any languages and mappings[11], as well as logics and serialisations (i.e. concrete syntaxes of languages).[12] The LoLa ("logics and languages") ontology formalises these notions [25]. LoLa and its main instance, the registry, form themselves a distributed ontology. The registry is written in RDF, LoLa in OWL plus some Common Logic axioms.

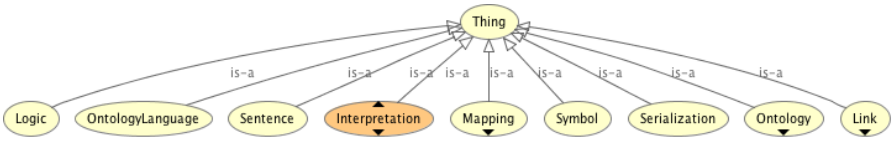


Fig. 3. Top-level classes in LoLa's OWL module

Figure 3 shows the top-level classes of LoLa's OWL module, axiomatising logics, languages, and mappings. Object-level classes (that is, classes providing the vocabulary for expressing distributed ontologies) comprise ontologies, their constituents (namely symbols and sentences), as well as links between ontologies. Mappings are modelled as shown in Figure 1: by a hierarchy of properties corresponding to the different types of edges in Figure 2. The full LoLa ontology is available at `http://purl.net/dol/1.0/rdf#`.

[11] As distributed ontologies refer to languages and mappings by IRIs, third parties may also set up their own, decentral registry extensions.

[12] The OWL 2 DL language is, e.g., exactly as expressive as the logic $\mathcal{SROIQ}(D)$ [17], and it can be serialised in the text-based Manchester syntax or as XML.

3 The Language DOL

3.1 Motivation

Many (domain) ontologies are written in DLs such as $\mathcal{SROIQ}$ and its profiles. These logics are characterised by having a rather fine-tuned expressiveness, exhibiting (still) decidable satisfiability problems, whilst being amenable to highly optimised implementations.

However, expressiveness beyond standard DLs is required for many foundational ontologies (as well as bio-medical ontologies), for instance DOLCE[13], BFO[14], or GFO[15]. Moreover, for practical purposes, these foundational ontologies also come in different versions ranging in expressiveness, typically between OWL (e.g. DOLCE Light, BFO-OWL) and first-order (DOLCE, GFO) or even second-order logic (BFO-Isabelle).

The relation between such different versions, OWL and first-order, may be recorded in various ways. In some cases it is primarily discussed in the research literature, see Keet's mereo-topological ontology [18] for an example, or it is described in the OWL ontology within a comment, not carrying formal semantics. Such a comment might only contain an *informal explanation* of how the OWL approximation was obtained (DOLCE Light is an example), but it might also describe a fully formal, axiomatised first-order extension of the OWL ontology.

Consider the BFO-OWL object property *temporalPartOf*. The OWL axiomatisation states this to be a transitive subproperty of *occurrentPartOf*, and the inverse of *hasTemporalPart*.[16] This property is, however, annotated in a rich way, containing example usages, a richer first-order axiomatisation of this property with pointers to the corresponding axioms in the first-order version, as well as natural language renderings of these axioms. The logical part of this annotation may be captured in DOL as follows: an OWL ontology first lists the entire OWL axiomatisation of BFO. In a second step, we *import* this OWL ontology along a translation to Common Logic, and subsequently extend the resulting first-order version of BFO-OWL with the first-order axioms previously only listed as comments. We obtain a two-level specification of BFO: the original OWL part (supported by OWL reasoners) and the full first-order part in Common Logic (amenable to first-order theorem proving and non-conservatively extending the OWL consequences).

3.2 DOL Syntax and Semantics

The DOL language is not "yet another ontology language", but a *meta language* for expressing relations between ontologies. Therefore, any ontology written in any conforming ontology language also is a DOL ontology. This has the clear

[13] See `http://www.loa.istc.cnr.it/DOLCE.html`

[14] See `http://www.ifomis.org/bfo/`

[15] See `http://www.onto-med.de/ontologies/gfo/`

[16] Parthood, typically understood as an anti-symmetric relation in mereology, is the canonical example of a relation that cannot be adequately formalised in OWL; a corresponding comment can be found in many bio-medical ontologies.

advantage that users can leave their ontologies as they are when working with DOL.

DOL provides two main abstract syntax categories:

1. *Modular* and *heterogeneous* ontologies. Such an ontology is written in a modular way, with the help of structuring operations. The semantics of ontologies is given by a signature and a class of models. In some cases, we can additionally provide a theory-level semantics of ontologies, as a signature and a class of sentences that, if it exists, agrees with the model-level semantics (that is, the model class is equal to the class of models satisfying the theory). We call an ontology *flattenable* if it has a theory-level semantics and *elusive* if it only admits a model-level semantics. This can be decided according to the outermost structuring operation on ontologies, as follows:

   Flattenable ontologies: basic ontologies, extension, union, translation, interpolate/forget, extract, reference, qualification, combination, bridge. Among these operations, interpolate/forget and extract can only be applied to flattenable ontologies.

   Elusive ontologies: reduction, minimisation and maximisation.

   For detailed definitions of these types of ontologies, see Section 3.3.
2. *Distributed* ontologies. These consist of of a list of declarations involving (possibly modular and/or heterogeneous) ontologies. These declarations can be ontology definitions (assigning a name to an ontology), interpretations (specifying a logical consequence relationship between ontologies), equivalences of ontologies (specifying that their model classes are in bijective correspondence), module relations (between ontologies and their modules), ontology alignments, and qualifications of the language, logic and/or serialisation. This will be detailed in Section 3.4.

3.3 Modular and Heterogeneous Ontologies

A (possibly modular and/or heterogeneous) ontology can be one of the following:

(a) a *basic ontology* O written inline, in a conforming ontology language and serialisation. The semantics is inherited from the ontology language. O can also be an ontology fragment, which means that some of the symbols or axioms may refer to symbols declared outside O (i.e. in an imported ontology). This is mainly used for extensions and equivalences. Here are two sample ontologies in OWL (using Manchester syntax) and Common Logic (using CLIF):

```
Class: Woman EquivalentTo: Person and Female
ObjectProperty: hasParent
```

```
(cl-module PreOrder
  (forall (x) (le x x))
  (forall (x y z) (if (and (le x y) (le y z)) (le x z))))
```

(b) an ontology qualified with the ontology *language* that is used to express it (written **language** $l : O$, where l identifies a language). Similarly, qualifications can also be by *logic* (written **logic** $l : O$), and/or *serialisation* (written **syntax** $s : O$).[17]

(c) an IRI reference to an ontology existing on the Web[18], possibly abbreviated using prefixes.[19] For example:

```
%prefix(
  co-ode: <http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/> )%
http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/pizza.owl
co-ode:pizza.owl
```

(d) an *extension* of an ontology by new symbols and axioms, written O_1 **then** O_2, where O_2 is an ontology (fragment) in a conforming ontology language. The resulting signature is that of O_1, augmented with the symbols in O_2. A model of an extension ontology is a model of this signature, that satisfies the axioms on O_2 and is (when appropriately reduced) a model of O_1. An extension can optionally be marked as conservative (%mcons or %ccons after the "**then**"). The semantics is that each O_1-model must have at least one expansion to the whole extension O_1 **then** O_2 (for %mcons) resp. that each logical consequence of O_1 **then** O_2 is already one of O_1 if it is over the signature of O_1 (for %ccons). In case that O_2 does not introduce any new symbols, the keyword %implied can be used instead of %ccons or %mcons; the extension then merely states intended logical consequences. The keyword %def stands for definitional extensions. This is similar to %mcons, but the model expansion must always exist uniquely. The following OWL ontology is an example for the latter:

```
  Class Person
  Class Female
then %def
  Class: Woman EquivalentTo: Person and Female
```

(e) a *union* of two self-contained ontologies (not fragments), written O_1 **and** O_2. Models of this union are those models that are (perhaps after appropriate reduction) models of both O_1 and O_2. For example, the class of commutative monoids can be expressed as

```
algebra:Monoid and algebra:Commutative
```

Forming a union of ontologies is a particularly common operation in the RDF logic, where it is known as merging graphs [15, section 0.3]; however, the RDF language provides no explicit syntax for this operation. When multiple RDF ontologies ("graphs") contain statements about the same symbol ("resource"), i.e., syntactically, triples having the same subject, the effect

[17] Some of the following listings omit obvious qualifications for readability.

[18] Note that not all ontologies can be downloaded by dereferencing their IRIs. Implementing a catalogue mechanism in DOL-aware applications might remedy this problem.

[19] Some of the following listings abbreviate IRIs using prefixes but omit the prefix bindings for readability.

is that in the merged graph the resource will have all properties that have previously been stated about it separately. Different kinds of properties, e.g. multilingual labels, geodata, or outgoing links to external graphs, are often maintained in different RDF graphs, which are then merged; consider the following excerpt:

```
{ :UniBremen rdfs:label "Université de Brême"@fr . } and
{ :UniBremen geo:lat "53.108612"^^xsd:float . } and
{ :UniBremen owl:sameAs[20]
    <http://dbpedia.org/page/University_of_Bremen> . }
```

(f) a *translation* of an ontology to a different signature (written O **with** σ, where σ is a signature morphism) or into some ontology language (written O **with translation** ρ, where ρ is an institution comorphism). For example, we can combine an OWL ontology with a first-order axiom (formulated in Common Logic) as follows:

```
ObjectProperty: isProperPartOf
  Characteristics: Asymmetric
  SubPropertyOf: isPartOf
with translation trans:SROIQtoCL
then
  (if (and (isProperPartOf x y) (isProperPartOf y z)) (isProperPartOf x z))
```

Note that OWL can express transitivity, but not together with asymmetry.

(g) a *reduction* of an ontology to a smaller signature Σ is written O **reveal** Σ. Alternatively, it can be written O **hide** Σ, where Σ is the set of symbols to be hidden (i.e. this is equivalent to O **reveal** $\mathsf{Sig}(O) \setminus \Sigma$). The effect is an existential quantification over all hidden symbols. For example, when specifying a group in sorted first-order logic, using the CASL language,

```
sort Elem
ops 0: Elem; __+__: Elem * Elem -> Elem; inv: Elem -> Elem
forall x,y,z . 0 + x       = x
             . x + (y + z) = (x + y) + z
             . x + inv(x)  = 0
reveal Elem, 0, __+__
```

revealing everything except the inverse operation `inv` results in a specification of the class of all monoids that can be extended with an inverse operation, i.e. the class of all groups with inverse left implicit.
Here is an example of hiding:

```
ontology Pizza = %% a simplified remake of the Pizza ontology [16]
  Individual: TomatoTopping
  Individual: MozzarellaTopping DifferentFrom: TomatoTopping
  ObjectProperty: hasTopping
  Class: VegetarianTopping
    EquivalentTo: { TomatoTopping, MozzarellaTopping, ... }
```

[20] While *owl:sameAs* is borrowed from the *vocabulary* of OWL, it is commonly used in the RDF logic to link to resources in external graphs, which should be treated as if their IRI were the same as the subject's IRI.

```
  Class: VegetarianPizza SubClassOf: some hasTopping VegetarianTopping
  ...
end

ontology Pizza_hide_VegetarianTopping =
  Pizza hide VegetarianTopping
end
```

A reduction to a less expressive logic is written O **hide along** μ, where μ is an institution morphism. This is a common operation in TBox/ABox settings, where an ontology in an expressive language provides the terminology (TBox) used in assertions (ABox) stated in a logic that is less expressive but scales to larger data sets; OWL DL (whose logic is $\mathcal{SROIQ}$) vs. RDF is a typical language combination:

```
ontology TBoxABox =
  Pizza hide along trans:SROIQtoRDF
  then language lang:RDF syntax ser:RDF/Turtle : {
    :myPizza :hasTopping
      [ a :TomatoTopping ], [ a :MozzarellaTopping ] .
  }
```

(h) an *interpolation* of an ontology, either in a subsignature or a sublogic, optionally with respect to a logic L (written O **keep in** Σ **with** L, where Σ is a signature or a logic and L is a logic)[21]. The effect is that sentences not expressible in Σ are weakened or removed, but the resulting theory still has the same L-consequences. The "**with** L" is optional, it defaults to the logic of O. Technically, this is a uniform interpolant [41, 29]. In case that Σ is a sublogic, this is also called approximation [28]. For example, we can interpolate the first-order DOLCE mereology in OWL:[22]

```
DOLCE_Mereology keep in log:OWL
```

Dually, O **forget** Σ **with** L interpolates O with the signature $\mathsf{Sig}(O) \setminus \Sigma$, i.e. Σ specifies the symbols that need to be left out. Cf. the notion of forgetting in [41, 29]. For example,

```
  Pizza forget VegetarianTopping
```

This has a theory-level semantics, i.e. yields a theory in the reduced signature (without `VegetarianTopping`). By contrast `Pizza hide VegetarianTopping` has a model-level semantics.

(i) a module *extracted* from an ontology, written O **extract** c Σ **with** m. Here, Σ is a restriction signature (which needs to be a subsignature of $\mathsf{Sig}(O)$), c is one of %mcons and %ccons, and m identifies a module extraction method. The extracted module is a subontology of O with signature larger than (or equal to) Σ, such that O is a conservative extension of the extracted module.

[21] It is also possible to specify a signature and a logic simultaneously: O **keep in** Σ, L_1 **with** L_2

[22] Interpolants need not always exist, and even if they do, tools might only be able to approximate them.

Dually, O **remove** c Σ **with** m extracts w.r.t. the signature $\mathsf{Sig}(O) \setminus \Sigma$.[23] For example, using the syntactic locality-* extraction method [38]:

```
Pizza remove %mcons
  VegetarianTopping
  with <http://example.org/onto/module/syntactic-locality-*>
```

Table 1 illustrates some of the connections between (g)–(i). We have three ways of removing the class `VegetarianTopping` from the ontology `Pizza`: (1) using hiding, we keep the model class of `Pizza`, but just remove the interpretation of `VegetarianTopping` from each model. Note that the resulting ontology has

```
VegetarianPizza SubClassOf:
  Annotations: dol:iri (*)
  some hasTopping { TomatoTopping, MozzarellaTopping, ... }
```

as a logical consequence. This is also a consequence of the corresponding uniform interpolant

```
  Pizza forget VegetarianTopping
```

which captures the theory of `Pizza hide VegetarianTopping`. Note that there is a subtle difference between (model-theoretic) hiding and (consequence-theoretic) forgetting: a model satisfying the *theory* of O **hide** Σ might itself not be a model of O **hide** Σ. In examples involving "**with** L", the uniform interpolant can be weaker than the hiding, because it is only required to have the same logical consequences in some language L, and a formula like (*) might not be a formula of L. Finally, an extracted module does not contain (*), because it only selects a subontology, and `Pizza` does not contain (*). Note that while forget/keep and hide/reveal both work w.r.t. smaller signatures and sublogics, remove/extract does not work for sublogics. This is because remove/extract must always respect the conservative extension property, which may not be possible when projecting to a sublogic. And if conservativity cannot be guaranteed, then forget/keep can be used in any case.

(j) a *combination* of ontologies, written **combine** $O_1, \ldots, O_n$ $L_1, \ldots, L_m$. Here the L_j are links between ontologies, see below. For disambiguating the symbols in the combined ontology, the individual ontologies can be prefixed with labels, like $n : O$, which are scoped to the current distributed ontology. The simplest example of a combination is a disjoint union (we here translate OWL ontologies into many-sorted OWL in order to be able to distinguish between different universes of individuals):

```
ontology Publications1 =
  Class: Publication
  Class: Article SubClassOf: Publication
  Class: InBook SubClassOf: Publication
```

[23] Note that the resulting module can still contain symbols from Σ, because the resulting signature may be enlarged.

	remove/extract	forget/keep	hide/reveal
semantic background	conservative extension	uniform interpolation	model reduct
relation to original	subtheory	interpretable	interpretable
approach	theory level	theory level	model level
type of ontology	flattenable	flattenable	elusive
signature of result	$\geq \Sigma$	$= \Sigma$	$= \Sigma$
change of logic	not possible	possible	possible

Table 1. Extract – Forget – Hide

```
  Class: Thesis  SubClassOf: Publication
  ...

ontology Publications2 =
  Class: Thing
  Class: Article SubClassOf: Thing
  Class: BookArticle SubClassOf: Thing
  Class: Publication SubClassOf: Thing
  Class: Thesis  SubClassOf: Thing
  ...

ontology Publications_Combined =
combine
  1 : Publications1 with translation trans:OWL2MS-OWL,
  2 : Publications2 with translation trans:OWL2MS-OWL
  %% implicitly: Article ↦ 1:Article ...
  %%             Article ↦ 2:Article ...
end
```

(This example will be continued using bridges below.) If links or alignments are present, the semantics of a combination is a quotient of a disjoint union (aligned symbols are identified). Technically, this is a colimit, see [42, 6]. An example for this is given along with the examples for alignments below.

(k) a *minimisation* of an ontology imposes a closed-world assumption on part of the ontology. It forces the non-logical symbols declared in O to be interpreted in a minimal way. This is written **minimize { O }**. Symbols declared before the minimised part are considered to be fixed for the minimisation (that is, we minimise among all models with the same reduct). Symbols declared after the minimisation can be varied. This is borrowed from circumscription [26, 3]. Alternatively, the non-logical symbols to be minimised and to be varied can be explicitly declared: O **minimize** Σ_1 **vars** Σ_2. For example, in the following OWL theory, B2 is a block that is not abnormal, because it is not specified to be abnormal, and hence it is also on the table.

```
  Class: Block
  Individual: B1 Types: Block
```

```
  Individual: B2 Types: Block DifferentFrom: B1
then minimize {
        Class: Abnormal
        Individual: B1 Types: Abnormal }
then
  Class: OnTable
  Class: BlockNotAbnormal EquivalentTo:
    Block and not Abnormal SubClassOf: OnTable
then %implied
  Individual: B2 Types: OnTable
```

Dually to minimisations, there are also maximisations.

(l) an ontology *bridge*, written O_1 **bridge with translation** t O_2, where t is a logic translation. The semantics is that of O_1 **with translation** t **then** O_2. Typically, t will translate a language like OWL to some language for distributed description logic or $\mathcal{E}$-connections [4, 21, 8], and O_2 introduces some axioms involving the relations (introduced by t) between ontologies in O_1. For example,

```
Publications_Combined
bridge with translation trans:MS-OWL2DDL
  %% implicitly added by translation trans:MS-OWL2DDL:
  %% binary relation providing the bridge
  1:Publication ⊑→ 2:Publication
  1:PhdThesis ⊑→ 2:Thesis
  1:InBook ⊑→ 2:BookArticle
  1:Article ⊑→ 2:Article
  1:Article ⊒→ 2:Article
end
```

3.4 Distributed Ontologies

Distributed ontologies. These have an optional identifier, declared with **distributed ontology** *Id*, and consist of

(a) *ontology* definitions, written **ontology** $Id = O$. For example,

```
ontology co-code:Pizza =
  Class: VegetarianPizza
  Class: VegetableTopping
  ObjectProperty: hasTopping
  ...
end
```

(b) theory *interpretations*, written **interpretation** $Id : O_1$ **to** $O_2 = \sigma$, expressing that the σ-reduct of each model of O_2 is a model of O_1. Instead of σ, an institution comorphism can be referred to. For example, we can express that the natural numbers are a total order as follows:

```
interpretation i : TotalOrder to Nat = Elem ↦ Nat
```

Here is a more complex example in Common Logic from the COLORE repository [7]:

```
interpretation geometry_of_time %mcons :
 %% Interpretation of linearly ordered time intervals...
  int:owltime_le
 %% ... that begin and end with an instant as lines
 %% that are incident with linearly ...
  to { ord:linear_ordering and bi:complete_graphical
     %% ... ordered points in a special geometry, ...
       and  int:mappings/owltime_interval_reduction }
  = int:ProperInterval ↦ int:Interval end
```

(c) ontology *equivalences*, written **equivalence** $Id : O_1 \leftrightarrow O_2 = O_3$ **along** ρ_1, ρ_2, expressing that O_1 and O_2 have model classes that are in bijective correspondence. This is done by providing a (fragment) ontology O_3 such that $\rho_i(O_i)$ **then** O_3 is a definitional extension [22]. ρ_1 and ρ_2 are optional institution comorphisms that default to the identity. For example, Boolean algebras are equivalent to Boolean rings:

```
equivalence e : algebra:BooleanAlgebra ↔ algebra:BooleanRing =
∀ x,y
. x ∧ y = x*y
. x ∨ y = x + y + x*y
. ¬x = 1 + x
. x*y = x ∧ y,
. x+y = (x ∨ y) ∧ ¬(x ∧ y).
end
```

(d) *module* relations, written **module** $Id\ c : O_1$ **of** O_2 **for** Σ. This expresses that O_1 is a module of O_2 with restriction signature Σ and conservativity c. If c is %mcons, this means that every Σ-reduct of an O_1-model can be expanded to an O_2-model. If c is %ccons, this means that every Σ-sentence φ following from O_2 already follows from O_1. This relation shall hold for any module O_1 extracted from O_2 using the **extract** construct.

(e) *alignment* definitions, written **alignment** $Id\ card_1\ card_2 : O_1$ **to** $O_2 = c_1, \ldots, c_n$, where $card_1$ resp. $card_2$ specify constraints on the alignment relation concerning the source resp. target. Each $card_i$ is one of 1, ?, +, * ('1' for injective and total, '+' for total, '?' for injective and '*' for none). The c_j are correspondences of form $sym_1\ rel\ conf\ sym_2$. Here, sym_i is a symbol from O_i, *rel* is one of the built-in relations $>$, $<$, $=$, %, $\ni$, $\in$, $\mapsto$, or an identifier of a relation specified externally, and *conf* is an (optional) confidence value between 0 and 1. This syntax of alignments follows the Alignment API [9].[24] Alignments have no formal semantics, but they can be used in combinations. For example,

```
%prefix( :      <http://www.example.org/alignment#>
```

[24] Note that BioPortal's [37] mappings are correspondences in the sense of the Alignment API and hence of DOL. BioPortal only allows users to collect correspondences, but not to group them into alignments. In a sense, for each pair of ontologies, all BioPortal users contribute to a big alignment between these.

```
         lang:  <http://purl.net/dol/languages/>
         ser:   <http://purl.net/dol/serializations/>
         trans: <http://purl.net/dol/translations/> )%

distributed ontology Alignments

language lang:OWL2/DL syntax ser:OWL2/Manchester

alignment Alignment1 : { Class: Woman } to { Class: Person } =
  Woman < Person
end

ontology AlignedOntology1 =
  combine Alignment1
end

ontology Onto1 =
  Class: Person
  Class: Woman SubClassOf: Person
  Class: Bank
end

ontology Onto2 =
  Class: HumanBeing
  Class: Woman SubClassOf: HumanBeing
  Class: Bank
end

alignment VAlignment : Onto1 to Onto2 =
  Person = HumanBeing,
  Woman = Woman
end

ontology VAlignedOntology =
  combine 1 : Onto1, 2 : Onto2, VAlignment
  %% 1:Person is identified with 2:HumanBeing
  %% 1:Woman is identified with 2:Woman
  %% 1:Bank and 2:Bank are kept distinct
end

ontology VAlignedOntologyRenamed =
  VAlignedOntology with 1:Bank ↦ RiverBank, 2:Bank ↦ FinancialBank,
                        Person_HumanBeing ↦ Person
end
```

(f) qualifications choosing the ontology *language*, *logic*, and/or *serialisation*. This is written **language** *Id*, **logic** *Id* and/or **syntax** *Id*, referring to entries of a registry as explained in Section 2.3, and affects the subsequent definitions in the distributed ontology.

This completes our overview of DOL. The full syntax and semantics of DOL will be available at `wiki.ontohub.org` and later submitted to OMG for standardisation.

Note that we have not covered the role of annotations in DOL so far. For structured annotation of ontologies and their parts, e.g. with metadata, or possibly with ontological relations not built into DOL's syntax, DOL does not provide its own syntax, but relies on the existing RDF standard. DOL allows for giving identifiers to all entities of distributed ontologies and basic ontologies[25] and thus *enables* their annotation. Annotations can be maintained in an RDF ontology that is a part of the distributed ontology.

4 The Ontology Repository Ontohub

Ontohub (see `http://ontohub.org`) is a web-based repository engine for ontologies that are written either in DOL or in some specific ontology language.[26]

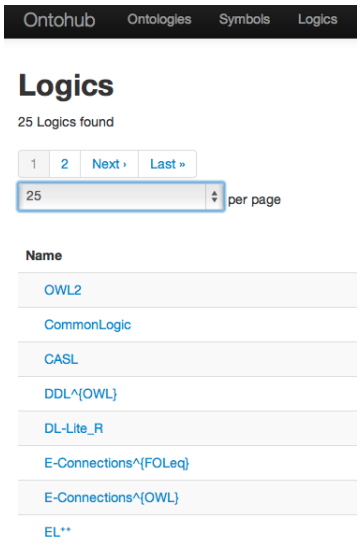


Fig. 4. `ontohub.org`: overview of logics

Ontohub provides means for organising ontologies into repositories. The distributed nature enables communities to share and exchange their contributions easily. The heterogeneous nature makes it possible to integrate ontologies written in various ontology languages. Ontohub supports a wide range of DOL-conforming ontology languages building on DOL and also supports DOL's interpretations, equivalences and alignments. Users of Ontohub can upload, browse, search and annotate single and distributed ontologies in various languages via a web front end. Figure 4 shows an excerpt of the 25 logics currently available in Ontohub.

The parsing and inference back end is the Heterogeneous Tool Set (Hets [31, 36], available at `hets.dfki.de`). Hets supports a large number of basic ontology languages and logics, as well as the DOL meta language

[25] When a basic ontology language has no mechanism for annotating or assigning identifiers to some ontology entities (as with imports in OWL or sentences in Common Logic), DOL provides a special comment syntax for injecting identifiers into basic ontologies written inline. Where identifiers in a basic ontology language are not IRIs, DOL allows for making them accessible as IRIs.

[26] Ontohub's sources are freely available at `https://github.com/ontohub/ontohub`.

as described in this paper.[27] The structural information extracted from DOL ontologies by Hets is stored in the Ontohub database and exposed to human users via a web interface and to machine clients as linked data.[28]

5 Conclusion and Future Work

The Distributed Ontology, Modelling and Specification Language (DOL) integrates different lines of research that have been reflected in the WoMO community (see [13, 12, 39, 19, 24, 40]):

- conservative extensions,
- ontology module extraction,
- ontology alignments,
- combinations of ontologies along alignments,
- distributed description logics,
- $\mathcal{E}$-connections, and
- relations between ontologies written in different languages (e.g. OWL and FOL).

DOL provides a unified meta language for these (and more) concepts, with a clean formal semantics. Tool support is provided by the Heterogeneous Tool Set (Hets) and by ontohub.org. The latter will also be used for the FOIS 2014 ontology competition. Since ontologies used in FOIS papers often need expressiveness beyond OWL, the multi-logic nature of DOL and Ontohub is essential.

A number of open problems and challenges remain:

- What is a suitable abstract meta framework for non-monotonic logics and rule languages such as RIF and RuleML? Are institutions suitable here? Are the modularity questions for these languages different from those for OWL?
- What is a useful abstract notion of ontology query (language)? How to handle answer substitutions in a logic-agnostic way?
- Can the notions of class hierarchy and of satisfiability of a class be generalised from OWL to other languages?
- Can logical frameworks be used for the specification of ontology languages and translations?

Acknowledgements

The development of DOL is supported by the German Research Foundation (DFG), Project I1-[OntoSpace] of the SFB/TR 8 "Spatial Cognition". Mihai Codescu was supported by the DFG, project SCHR1118-7-1. Christoph Lange was supported by EPSRC grant EP/J007498/1. The authors would like to

[27] Some (but only few) of DOL's features are still being implemented at the time of the writing of this paper.

[28] "Linked data" is a set of best practises for publishing structured data on the Web in a machine-friendly way [1]. DOL and Ontohub conform with linked data.

thank the OntoIOp working group for their valuable input, particularly Michael Grüninger, Maria Keet, Fabian Neuhaus and Peter Yim. We also want to thank Carsten Lutz and Thomas Schneider for valuable input on interpolation and module extraction.

References

1. Berners-Lee, T. Design Issues: Linked Data. July 27, 2006. http://www.w3.org/DesignIssues/LinkedData.html (visited on 2010-01-20).
2. Bidoit, M. and Mosses, P. D. *CASL User Manual*. LNCS (IFIP Series) 2900. Freely available at http://www.cofi.info. Springer, 2004.
3. Bonatti, P. A., Lutz, C., and Wolter, F. The Complexity of Circumscription in DLs. In *J. Artif. Intell. Res. (JAIR) 35* (2009), pp. 717–773.
4. Borgida, A. and Serafini, L. Distributed Description Logics: Assimilating Information from Peer Sources. In *Journal of Data Semantics 1* (2003), pp. 153–184.
5. Borzyszkowski, T. Logical systems for structured specifications. In *Theoretical Computer Science 286* (2002), pp. 197–245.
6. Codescu, M. and Mossakowski, T. Heterogeneous colimits. In *MoVaH'08 Workshop on Modeling, Validation and Heterogeneity*. Ed. by F. Boulanger, C. Gaston, and P.-Y. Schobbens. IEEE press, 2008. http://www.computer.org/portal/web/csdl/abs/proceedings/icstw/2008/3388/00/3388toc.htm.
7. COLORE. An open repository of first-order ontologies represented in Common Logic. http://colore.googlecode.com.
8. Cuenca Grau, B., Parsia, B., and Sirin, E. Ontology Integration Using $\mathcal{E}$-connections. In *Modular Ontologies—Concepts, Theories and Techniques for Knowledge Modularization*. Ed. by H. Stuckenschmidt, C. Parent, and S. Spaccapietra. Vol. 5445. LNCS. Springer, 2009.
9. David, J., Euzenat, J., Scharffe, F., and dos Santos, C. T. The Alignment API 4.0. In *Semantic Web 2*, 1 (2011), pp. 3–10.
10. Goguen, J. A. and Burstall, R. M. Institutions: Abstract Model Theory for Specification and Programming. In *Journal of the Association for Computing Machinery 39* (1992). Predecessor in: LNCS 164, 221–256, 1984., pp. 95–146.
11. Goguen, J. and Roşu, G. Institution morphisms. In *Formal aspects of computing 13* (2002), pp. 274–307.
12. Grau, B. C., Honavar, V., Schlicht, A., and Wolter, F., eds. Proceedings of the 2nd International Workshop on Modular Ontologies, WoMO 2007, Whistler, Canada, October 28, 2007. Vol. 315. CEUR Workshop Proceedings. CEUR-WS.org, 2008.
13. Haase, P., Honavar, V., Kutz, O., Sure, Y., and Tamilin, A., eds. Proceedings of the 1st International Workshop on Modular Ontologies, WoMO'06, co-located with the International Semantic Web Conference, ISWC'06 November 5, 2006, Athens, Georgia, USA. Vol. 232. CEUR Workshop Proceedings. CEUR-WS.org, 2007.
14. The NeOn Ontology Engineering Toolkit. http://www.neon-project.org/. 2008. http://watson.kmi.open.ac.uk/Downloads%20and%20Publications_files/neon-toolkit.pdf.
15. Hayes, P. RDF Semantics. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 10, 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

16. Horridge, M. Protégé OWL Tutorial. Version v1.3. Mar. 24, 2011. http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/.
17. Horrocks, I., Kutz, O., and Sattler, U. The Even More Irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*. AAAI Press, June 2006, pp. 57–67.
18. Keet, C. M. and Artale, A. Representing and reasoning over a taxonomy of part–whole relations. In *Applied Ontology 3*, 1 (2008), pp. 91–110.
19. Kutz, O., Hois, J., Bao, J., and Cuenca Grau, B., eds. *Modular Ontologies—Proceedings of the Fourth International Workshop (WoMO 2010)*. Vol. 210. Frontiers in Artificial Intelligence and Applications. Toronto, Canada: IOS Press, 2010.
20. Kutz, O., Lücke, D., Mossakowski, T., and Normann, I. The OWL in the CASL—Designing Ontologies Across Logics. In *OWL: Experiences and Directions, 5th International Workshop (OWLED-08)*. Ed. by C. Dolbear, A. Ruttenberg, and U. Sattler. co-located with ISWC-08, Karlsruhe, Germany, October 26–27: CEUR-WS, Vol-432, 2008.
21. Kutz, O., Lutz, C., Wolter, F., and Zakharyaschev, M. $\mathcal{E}$-connections of Abstract Description Systems. In *Artificial Intelligence 156*, 1 (2004), pp. 1–73.
22. Kutz, O., Mossakowski, T., and Lücke, D. Carnap, Goguen, and the Hyperontologies: Logical Pluralism and Heterogeneous Structuring in Ontology Design. In *Logica Universalis 4*, 2 (2010). Special issue on 'Is Logic Universal?'
23. Kutz, O., Lange, C., Mossakowski, T., Keet, C. M., Neuhaus, F., and Grüninger, M. The Babel of the Semantic Web Tongues – In Search of the Rosetta Stone of Interoperability. In *What will the Semantic Web look like 10 Years from now? Workshop at ISWC*. Ed. by F. van Harmelen, J. A. Hendler, P. Hitzler, K. Janowicz, and D. Vrandečić. 2012. http://stko.geog.ucsb.edu/sw2022/.
24. Kutz, O. and Schneider, T., eds. *Modular Ontologies—Proceedings of the Fifth International Workshop (WoMO 2011)*. Vol. 230. Frontiers in Artificial Intelligence and Applications. IOS Press, 2011.
25. Lange, C., Mossakowski, T., and Kutz, O. LoLa: A Modular Ontology of Logics, Languages, and Translations. In. Ed. by T. Schneider and D. Walther. Vol. 875. CEUR-WS, 2012. http://ceur-ws.org/Vol-875.
26. Lifschitz, V. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*. Vol. 3. Oxford University Press, 1994, pp. 297–352.
27. Lüttich, K., Masolo, C., and Borgo, S. Development of Modular Ontologies in CASL. In *WoMO*. Ed. by P. Haase, V. Honavar, O. Kutz, Y. Sure, and A. Tamilin. Vol. 232. CEUR Workshop Proceedings. CEUR-WS.org, 2006.
28. Lutz, C., Seylan, I., and Wolter, F. An Automata-Theoretic Approach to Uniform Interpolation and Approximation in the Description Logic EL. In *KR*. Ed. by G. Brewka, T. Eiter, and S. A. McIlraith. AAAI Press, 2012.
29. Lutz, C. and Wolter, F. Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics. In *IJCAI*. 2011, pp. 989–995.
30. Meseguer, J. General Logics. In *Logic Colloquium '87*. Ed. by H. J. Ebbinghaus. North Holland, 1989, pp. 275–329.
31. Mossakowski, T. Hets: the Heterogeneous Tool Set. http://hets.dfki.de (visited on 2012-12-10).
32. Mossakowski, T., Haxthausen, A., Sannella, D., and Tarlecki, A. CASL: The Common Algebraic Specification Language. In *Logics of Formal Specification Languages*. Ed. by M. H. D. Bjorner. Monographs in Theoretical Computer Science. Springer-Verlag Heidelberg, 2008. Chap. 3, pp. 241–298. http://dx.doi.org/10.1007/978-3-540-74107-7_5.

33. Mossakowski, T. and Kutz, O. The Onto-Logical Translation Graph. In *Modular Ontologies*. Ed. by O. Kutz and T. Schneider. IOS, 2011.
34. Mossakowski, T., Kutz, O., and Lange, C. Semantics of the distributed ontology language: Institutes and Institutions. In *Recent Trends in Algebraic Development Techniques, 21th International Workshop, WADT 2012*. Ed. by N. Martí-Oliet and M. Palomino. Vol. 7841. Lecture Notes in Computer Science. Springer, 2013, pp. 212–230. http://link.springer.com/chapter/10.1007/978-3-642-37635-1_13.
35. Mossakowski, T., Lange, C., and Kutz, O. Three Semantics for the Core of the Distributed Ontology Language. In *7th International Conference on Formal Ontology in Information Systems (FOIS)*. Ed. by M. Donnelly and G. Guizzardi. Vol. 239. Frontiers in Artificial Intelligence and Applications. FOIS Best Paper Award. IOS Press, 2012, pp. 337–352.
36. Mossakowski, T., Maeder, C., and Lüttich, K. The Heterogeneous Tool Set. In *TACAS 2007*. Ed. by O. Grumberg and M. Huth. Vol. 4424. Lecture Notes in Computer Science. Springer-Verlag Heidelberg, 2007, pp. 519–522.
37. Noy, N. F., Shah, N. H., Patricia L. Whetzel, ., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C. G., and Musen, M. A. BioPortal: ontologies and integrated data resources at the click of a mouse. In *Nucleic Acids Research 37* (2009). http://bioportal.bioontology.org, W170–W173.
38. Sattler, U., Schneider, T., and Zakharyaschev, M. Which Kind of Module Should I Extract? In *Proceedings 22nd Int. Workshop on Description Logics (DL)*. Vol. 477. CEUR Workshop Proceedings. CEUR-WS.org, 2009.
39. Sattler, U. and Tamilin, A., eds. Workshop on Ontologies: Reasoning and Modularity (WORM-08). Vol. Vol-348. (ESWC) Tenerife, Spain: CEUR Workshop Proceedings, 2008. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-348/.
40. Schneider, T. and Walther, D., eds. Proc. of the 6h Int. Workshop on Modular Ontologies. Vol. 875. CEUR-WS, 2012.
41. Wang, Z., Wang, K., Topor, R. W., and Pan, J. Z. Forgetting for knowledge bases in DL-Lite. In *Ann. Math. Artif. Intell. 58*, 1–2 (2010), pp. 117–151.
42. Zimmermann, A., Krötzsch, M., Euzenat, J., and Hitzler, P. Formalizing Ontology Alignment and its Operations with Category Theory. In *Proc. of FOIS-06*. 2006, pp. 277–288.

Combining Ontologies in Settings with Multiple Agents

George A. Vouros

Department of Digital Systems,
University of Piraeus, Greece
georgev@unipi.gr

Abstract. Combining knowledge and beliefs of autonomous peers in distributed settings, is a major challenge. In this talk we consider agents that combine their ontologies and reason jointly with their coupled knowledge using the E-SHIQ representation framework. We motivate the need for a representation framework that allows agents to combine their knowledge in different ways, maintaining the subjectivity of their own knowledge and beliefs, and to reason collaboratively, constructing a tableau that is distributed among them. The talk presents the $E-\mathcal{SHIQ}$ representation framework and the tableau reasoning algorithm. It presents the implications to the modularization of ontologies for efficient reasoning, implications to coordinating agents' subjective beliefs, as well as challenges for reasoning with ontologies in open and dynamic multi-agent systems.

1 Combining Ontologies with $E-\mathcal{SHIQ}$

To combine knowledge and beliefs of autonomous agents in open and inherently distributed settings, we need special formalisms that take into account the complementarity and heterogeneity of knowledge in multiple interconnected contexts. Agents may have different, subjective beliefs concerning "bridging" heterogeneity and coupling their knowledge with the knowledge of others. The subjectivity of beliefs plays an important role in such a setting, as agents may inherently (i.e. due to restrictions of their task environment) have different views of the knowledge possessed by others, or they may not agree on the way they may jointly shape knowledge.

On the other hand, large ontologies need to be dismantled so as to be evolved, engineered and used effectively during reasoning. The process of taking an ontology to possibly interdependent ontology units is called ontology modularization, and specifically, ontology partitioning. Each such unit, or module, provides a specific context for performing ontology maintenance, evolution and reasoning tasks, at scales and complexity that are smaller than that of the initial ontology. Therefore, in open and inherently distributed settings (for performing either ontology maintenance, evolution or reasoning tasks), several such ontology modules may co-exist in connection with each other. Formally, it is required that any axiom that is expressed using terms in the signature of a module and it is

entailed by the ontology must be entailed by the module, and vise-versa. The partitioning task requires that the union of all the modules, together with the set of correspondences/relations between modules, is semantically equivalent to the original ontology. This later property imposes hard restrictions to the modularization task: Indeed, to maintain it, a method must do this with respect to the expressiveness of the language used for specifying correspondences/relations between modules' elements, to the local (per ontology module) interpretation of constructs, and to the restrictions imposed by the setting where modules are deployed.

The expressivity of knowledge representation frameworks for combining knowledge in multiple contexts, and the efficiency of distributed reasoning processes, depend on the language(s) used for expressing local knowledge and on the language used for connecting different contexts.

While our main goal is to provide a rich representation framework for combining and reasoning with distinct ontology units in open, heterogeneous and inherently distributed settings, we propose the $E-\mathcal{SHIQ}$ representation framework and a distributed tableau algorithm [1] [2].

The representation framework $E-\mathcal{SHIQ}$:

- Supports subjective concept-to-concept correspondences between concepts in different ontology units.
- In conjunction to subjective concept-to-concept correspondences, $E-\mathcal{SHIQ}$ supports relating individuals in different units via link relations, as well as via subjective individual correspondence relations. While correspondence relations represent equalities between individuals, from the subjective point of view of a specific unit, link relations may relate individuals in different units via domain-specific relations.
- Supports distributed reasoning by combining local reasoning chunks in a peer-to-peer fashion. Each reasoning peer with a specific ontology unit holds a part of a distributed tableau, which corresponds to a distributed model.
- Finally, $E-\mathcal{SHIQ}$ inherently supports subsumption propagation between ontologies, supporting reasoning with concept-to-concept correspondences in conjunction to link relations between ontologies.

2 Constructing $E-\mathcal{SHIQ}$ distributed knowledge bases via modularization

To distribute knowledge among different agents, we need to partition monolithic ontologies to possibly interconnected modules. In this part of the talk we describe efforts towards constructing $E-\mathcal{SHIQ}$ distributed knowledge bases by modularizing ontologies: Our aim is to make ontology units as much self-contained and independent from others as possible, so as to increase the efficiency of the reasoning process. We discuss the flexibility offered by $E-\mathcal{SHIQ}$ itself, and different modularization options available (a first attempt towards this problem has been reported in [3]).

3 Challenges towards reasoning with multiple ontologies

Towards reasoning with ontology units in open and dynamic settings with multiple agents, this talk presents and discusses the following major challenges:

Reaching Agreements to correspondences: Agents in inherently distributed and open settings can not be assumed to share an agreed ontology of their common task environment. To interact effectively, these agents need to establish semantic correspondences between their ontology elements. As already pointed out, the correspondences computed by two agents may differ due to (a) different mapping methods used, to (b) different information one makes available to the other, or (c) restrictions imposed by their task environment. Although semantic coordination methods have already been proposed for the computation of subjective correspondences between agents, we need methods for communities, groups and arbitrarily formed networks of interconnected agents to reach semantic agreements on subjective ontology elements' correspondences [4].

Exploitation of ontology units in open and dynamic settings: In open settings where agents may enter or leave the system at will, we need agents to dynamically combine their knowledge and re-organize themselves, so as to form groups that can serve specific information needs successfully. There are several issues that need to be addressed here: Agents (a) must share information about their potential partners and must learn the capabilities, effectiveness, trustworthiness etc. of their peers, (b) must locate the potential partners, and (c) must decide for the "best" groups to be formed in an ad-hoc manner, towards serving the specific information needs. Reaching complete and optimal solutions in such a setting is a hard problem: we discuss the computation of approximate solutions [5].

Acknowledgements Thanks to Georgios Santipantakis for his contributions to various parts of this work, especially the one concerning $E - \mathcal{SHIQ}$. The major part of the research work referenced in this talk is being supported by the project IRAKLITOS II" of the O.P.E.L.L. 2007 - 2013 of the NSRF (2007 - 2013), co-funded by the European Union and National Resources of Greece.

References

1. Vouros, G.A., Santipantakis, G.M.: Distributed reasoning with $e^{DDL}_{HQ^+}\mathcal{SHIQ}$. In: Modular Ontologies: Proc. of the 6th International Workshop (WoMo 2012). (July 2012)
2. Santipantakis, G.M., Vouros, G.A.: The e-shiq contextual logic framework. In: AT. (2012) 300–301
3. Santipantakis, G.M., Vouros, G.A.: Modularizing owl ontologies using $e^{DDL}_{HQ^+}\mathcal{SHIQ}$. In: ICTAI. (2012) 411–418
4. Vouros, G.A.: Decentralized semantic coordination via belief propagation. In: AAMAS. (2013) 1207–1208
5. Karagiannis, P., Vouros, G.A., Stergiou, K., Samaras, N.: Overlay networks for task allocation and coordination in large-scale networks of cooperative agents. Autonomous Agents and Multi-Agent Systems **24**(1) (2012) 26–68

Modularization of Graph-Structured Ontology with Semantic Similarity

Soudabeh Ghafourian, Amin Rezaeian, and Mahmoud Naghibzadeh

Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
so.ghafourian@stu-mail.um.ac.ir
amin.rezaeian@stu-mail.um.ac.ir
naghibzadeh@um.ac.ir

Abstract. Modularization is a key requirement to manage the size and complexity of large ontologies by replacing each one by a set of smaller ontologies. Two reasons for this requirement are that current ontology languages such as OWL do not allow partial reuse of ontologies and ontologies are ever growing to cover more knowledge in a specific domain. Many existing modularization methods focus on either semantics or structural aspects of ontologies while both of them are important. In this paper, we consider both semantic and structure by combining these aspects using random walk algorithms to achieve a balance between them. We also define weights for different relations to take semantic into account. The proposed method is designed using two algorithms: a greedy algorithm and a heuristic one to reduce run-time and time-complexity. Our goal is to produce reusable modules of high quality and support large ontologies. The results of the experiments show that our algorithms perform well in comparison with existing golden standard.

Keywords: Ontology Modularization, Partitioning, Ontology Reuse, Semantic Similarity

1 Introduction

Today, we use ontologies in the context of information processing, semantic web, and many other contexts [1]. Large ontologies usually contain many terms. These large ontologies are confronted with challenges in their life cycle such as processing and maintenance [2].

Modularization is an approach to tackle challenges of large ontologies. An ontology module is a small ontology that can have inter-module links with other small ontologies [1] and the union of the produced modules is semantically equal to the first ontology [3].

Ontology modularization is used to achieve different goals such as scalability for querying data and reasoning on ontologies, scalability for evolution and maintenance, complexity management, understandability, context-awareness and personalization and reusability. The goals of ontology modularization can affect the understandability, the advantages, and disadvantages of the resulting modules [1].

Reusability is one of the important goals of modularization of ontologies that apply when designing ontologies or when developing new applications based on ontologies. Current ontology languages such as OWL do not allow importing only parts of an ontology. This is a problem because if an ontology developer needs to reuse only parts of an ontology, they must import the whole ontology, which takes more space [4]. For example, we have home appliances ontology and an ontology developer who is interested in small home appliances; the developer would like to import the part of the ontology, which concerns only small home appliances. Therefore, he is not interested in loading the whole ontology but rather an extracted module of the ontology [2].

In this paper, we present modularization algorithms that assign weights to the different relations that are used in the formalization of the ontology. The goal of our modularization is reusability. The proposed modularization algorithms are performed after the weighing phase of different relations is completed. Until now, some of these relations such as inverseOf, unionOf, intersectionOf, disjointWith, have not been considered. Since different kinds of the relations represent different semantic aspects of the ontology, the weighing is determined according to the importance of these relations. The proposed algorithms consider both structural and semantic criteria.

Our goal is to produce modules in which concepts are both structurally and semantically closely related; hence, the resulting subontologies can be reused in developing new applications.

Modularization is done based on an agglomerative hierarchical algorithm on which we perform some optimization to achieve a better result and use new scoring function. We also propose another algorithm to reduce time-complexity.

The remainder of the paper is organized as follows: in the next section, we briefly present some of the related work. In section 3, we define weights for different relations; describe our modularization algorithms and criteria for modularization. In Section 4, we evaluate the proposed method and report our results. Finally, Section 5 concludes this paper with future work suggestions.

2 Related work

In this section, we review related work to ontology modularization. Recently, many different approaches have been proposed for this purpose. There are several categories for modularization. We categorize modularization based on types of representation of ontology into two sets: graph-based approaches and logic-based approaches [2-3]. Our work is based on graph approaches. In logic-based approaches, the modules are produced based on logical representation of ontologies. The work in [5] extracts modules from OWL-DL ontology based on user's semantic query. Graph-based approaches use graph-theoretic algorithms to traverse the hierarchy of ontologies and some heuristics to get relevant modules [6]. In general, the works [7-10] use agglomerative algorithms to modularize ontologies. An agglomerative approach is an iterative bottom up one in which, in each iteration, two modules with the highest similarity are merged to produce a new module. The work in [7] defines structural and linguistic

similarities. The first criterion is based on hierarchy of classes and the other one is based on similarities between the local descriptions of the classes. They present partition algorithms that combine criteria as a scoring function. Their goal is ontology matching. In matching ontologies, one tries to find the most similar ontology amongst a set of ontologies to the one for which a match is requested. The approach performs modularization of each two ontologies whose similarity is needed. In the modularization, the hierarchical subclassOf relation and a linguistic similarity measure are used.

In [8], A weighted graph is constructed from rdfs:subClassOf and rdfs:subPropertyOf relationships. owl:equivalentClass or owl:equivalentProperty are identified in a preprocessing stage. They present a structure-oriented partitioning algorithm and add RDF sentences to construct blocks from the modules. The goal of this work is ontology matching. This work only considers limited relationships. Relations such as inverseOf, unionOf, intersectionOf, disjointWith, are not considered.

Paper [11] describes a structure-based ontology partitioning. They construct a directed weighted graph based on structure similarities. Five types of criteria between concepts, i.e., subclass, domain/range, definition, substring, and distance relation, are defined. The weighted matrix is constructed according to the number of connections between nodes; then they use Line Island Method [12] to partition and finally perform optimization to improve the partitioning.

The work in [13] proposes an ontology partitioning method, which produces overlapped modules, i.e., final modules may have common concepts. They use semantic similarity between concepts, so their generated graph is conceptual. This work considers structure and semantic, but it does not distinguish between different relationships.

In [9], different graph representations for ontologies are developed. There are three basic representations and two different extensions of these representations. They use several methods to convert ontologies into graphs, and apply community detection algorithms to partition the graphs. Their experiments show that the algorithms work much better when subject, object and predicate are represented as different nodes. Paper [10] further develops the findings of paper [9], but the main difference is that they define a weight function for different relations of the ontology as shown in Table 1; this is the first steps in a semantic approach. They apply three community detection algorithms (a type of an agglomerative algorithm) on different graph representations.

The general topic of papers [7-8] is ontology matching based on structural aspects of ontologies. However, we are interested in ontology modularization with the goal of reusability of resulting modules. The works described in [11] and [13] mostly use structure in order to make a graph representation of an ontology and use classic graph clustering methods to modularize ontologies, while our method uses a hierarchical clustering method. We use different graph clustering methods than [9-10], in addition we consider more relations. These relations are mentioned in section 3.2.

3 Proposed Approach

In this section, we first describe how we represent ontologies. Then we present how we construct the weighted matrix. Next, we normalize the weights of edges, and then Neighborhood Random-Walk Distance is introduced [15]. Finally, the modularization algorithms are described.

Our goal is to bring together into one module the most related concepts that have highest semantic similarities and presumably describe one subdomain. This agrees with the concept of domain specific ontologies [10]. If a good modularization algorithm such as agglomerative algorithms and a suitable scoring function is used this goal is reachable.

Agglomerative algorithms [16] are algorithms where the modules with the highest similarity are iteratively merged. They are bottom-up, this means, initially every node is considered as an independent module and in the end there is only one module.

3.1 Different Graph Representations of Ontology

Ontology web language[1] (OWL) is a semantic web language. It is based on the Resource Description Framework[2] (RDF). RDF represents information as triples of the form (Subject, Predicate, and Object).RDF triples can be mapped to a graph where subject and object are nodes and each predicate is a directed edge from a subject to an object. It displays a simple mental model for RDF that is frequently used [14].

We also use other graph models; since the predicate of one triple is a subject or an object in some other triples, we represent every subject, object, and predicate as separate nodes [10].

There are various representations of ontologies [10].We represent each one of subjects, objects and predicates as a separate node in which we have two types of edges: one type of edge is from subject to predicate and another is from predicate to object. The predicate node contains object or datatype properties. We also consider every individual as a node.

3.2 Constructing Weighted Matrix

We define a weight function to give weighs to different relationships of the ontology. This function assigns an integer number to existing relationships as shown in Table 1. The main reason for weighing relationships is that different relationships have different semantics and show different aspects of the ontology. We would like to distinguish between these aspects and their importance by their weights. This is not meant that a relation with a higher weight is more valuable than a relation with a lower weight, but sometimes it means that a relation with a higher weight has existential precedence over a relation with a lower weight. Therefore, the weights can be changed for different applications and/or in different contexts. For example, if sub-

[1] OWL - http://www.w3.org/OWL
[2] RDF - http://www.w3.org/RDF

ClassOf relation exists because it is part of ontology language, then domain and range relations are meaningful. Therefore, we assign weight 10 to subClassOf and weight 5 (i.e., one half the weight) to domain/range relation. In some other situations, the weights are assigned based on the wideness or narrowness of their meanings. Examples are given below.

List of Different Relationships and Weights. The weights represented in Table 1 are based on previous research and also our assessments of relations which are not studied by previous research. The base of weights is what is mentioned in [10]. For new relations, the weight assignment logic is explained in the previous subsection.

The equivalent relation denotes that the classes have the same meaning, so the classes that have equivalent relation are put into one module. Considering the meaning of the equivalent relation, they are given the highest weight. Furthermore, the subclass relations give some important information about classes; hence, these relationships have high semantic contribution to ontology modularization. As mentioned before, the weight of this relationship is higher than that of domain/range relationship but not as much as the equivalent relationship has [10].

Table 1. List of relationships and weights

Property	Weight	Property	Weight
equivalentClass	20 [10]	unionOf	10
subClassOf	10 [10]	intersectionOf	10
subPropertyOf	10 [10]	disjointWith	0-10
domain	5 [10]	complementOf	10
range	5 [10]	inverseOf	20
comment	0.2 [10]	FunctionalProperty	5
seeAlso	0.2 [10]	InverseFunctionalProperty	5
isDefinedBy	0.2 [10]	Other relations	1
label	0.2 [10]		

The union and intersection relations are the same as subclass relations because if for example class A is the union of C, B and D then each class C, B and D is a subClassOf A.

If disjoint relation exists between highest-level concepts, the weight is considered to be zero because they are really disjoint, however if it occurs in lowest-level concepts, the weight is considered 10. If it occurs somewhere in between, the weight is assigned accordingly.

When two concepts have a complement relationship, it means they are strongly connected. We consider that at first, they have a subclass relationship (their super class is the universal set) and then they have a complement relationship.

We put the inverseOf relations in the modules of the property that is related to, so its weight should be high.

For object property, when the properties have an inverse functional attribute, it means this property implies unique value and on the other hand, domain and range

edges represent subdomains of ontologies. We add the restrictions such as cardinality after modularization because they contain literal values.

Unifying the Weight of Edges. If there is more than one relationship between two nodes, we add up the weights of all the existing relationships between the nodes. Thus, all weights are taken into consideration in our modularization algorithms.

3.3 Normalization

In this phase, the weights of the edges in the graph are normalized to be between zero and one. Thus, the weight of the edge outgoing from a node v is divided by the sum of the weights of outgoing edges from node v. This is needed for input matrix of random walk in which every element must be between zero and one.

$$W_{i,v}^{normal} = \frac{\mathrm{w}_{i,v}}{\Sigma_{j \in out\_edges(V)} W_{j,v}} \quad (1)$$

Where $W_{i,v}$ is the weight of the edge outgoing from a node v that is normalizing and the denominator is the sum of the weights of outgoing edges from node v.

3.4 Neighborhood RandomWalk Distance

We use the neighborhood random walk method [15] to measure vertex closeness. A random walk is a mathematical representation of the path one may navigate through multiple random steps.

$$d(v_i, v_j) = \sum_{T:v_i \to v_j} P(T) c(1-c)^{Length(T)} \quad (2)$$

Where P is transition probability matrix, $Length(T)$ is length of random walk where $Length(T) \leq l$, l is the length that a random walk can go, c is restart probability where $c \in (0,1)$, $d(v_i, v_j)$ is the neighborhood random walk distance from v_i to v_j. It measures vertex closeness and T is a path from v_i to v_j whose length is $Length(T)$ with transition probability $P(T)$.

We perform matrix multiplication on a transition probability matrix of the ontology graph to use the neighborhood random walk model [15].

$$R^l = \sum_{\gamma=0}^{l} c(1-c)^{\gamma} P^{\gamma} \quad (3)$$

In this equation, R is the neighborhood random walk distance matrix. Intuitively every element at row i, column j in R, captures the probability of navigating from node i to j with at most l steps in graph. l is the length that a random walk can go and it comes from the previous formula, and γ is the random walk step. The neighborhood random walk distance matrix is constructed in the following steps:

1. Assigning weights to different relationships
2. Normalization of weight of step 1
3. Constructing transition probability and neighborhood random walk distance matrix

3.5 Modularization Algorithm

The proposed modularization algorithm in this section is an agglomerative algorithm. The main difference between our algorithm and other similar algorithms in [7-8] and [9-10] is that we use a new scoring function that calculates both intra-and inter-connectivity. We apply scoring function for every concept node in our algorithm in order to improve the efficiency. It means that the distance between the node and every other node is measured. Another difference is that we consider more relations than other approaches. We use an agglomerative algorithm, such that in each iteration, we select two modules that have the highest positive impact on the score of modularization. Then those modules are merged. From the scoring function perspective, the scores of other modules may not change. This way the required computation for computing modularization score is not high. The process is shown in Algorithm 1.

The advantages of agglomerative algorithms are that we don't have to know the size and the number of modules and the result of these algorithms depend on the chosen similarity criterion [16].The input to our algorithm is the neighborhood random walk distance matrix.

Our criterion function is the silhouette coefficient $s(i)$ [17] where $-1 \leq s(i) \leq 1$. If $s(i)$ is close to one it means that the node will be appropriately grouped. The average $s(i)$ of a module is a measure that shows how appropriately the nodes have been grouped into modules.

$$s(i) = \frac{a(i)-b(i)}{max\{a(i),b(i)\}} \tag{4}$$

Where i is node, $a(i)$ is the average similarity of i to all other nodes within the same module, $b(i)$ is the highest average similarity of i to nodes of other modules, and $s(i)$ should be computed for each node i. For the module c, s_c which is the average of all $s(i)$ for all nodes i in module c, is defined as follows:

$$s_c = \frac{\sum_{i \in c} s(i)}{n_c} \tag{5}$$

In this equation n_c is the number of nodes in module c. To score the modularization C, we define the scoring function as follows:

$$score(C) = average_{c \in C}(s_c) \tag{6}$$

Equation (6) is used to compute efficiency of the modularization. It comes from the average of scores of each module. Each module's score is computed by the average of scores of its nodes. As the score of every node is calculated using both intra- and inter-module connections, the resulting efficiency of modularization is effected by both intra- and inter-module connections of all nodes in graph.

```
Algorithm 1. Modularization (adjacencyMatrix)
 //C represents whole modularization
 C = put every node in a separate module
 for K=N down to 2   // K shows number of modules
```

```
// for all the pairs of nodes that could be merged
  maxS= -2
  for i=1 to K-1
    for j=i+1 to K
      c=Union(i,j);//merge modules i and j into module C
      C=C∪{c}\{i,j};
       S=Score(C); //according to equation (6)
       //if this new score is better, save it
      if S>maxS
        maxS = S;
        modularization=C;
      end if
    end for
  end for
  bestModularization=modularization;
end for
```

Heuristic Algorithm. We propose the heuristic algorithm to support large ontologies and reduce time-complexity. It is shown in Algorithm 2. The input of this algorithm is incidence matrix that is constructed from adjacency matrix R that is calculated using equation (3). Each row of incidence matrix represents an edge, which consists of first node, second node and weight. It is useful for large data sets. This matrix is ordered descending by weight column.

The time-complexity of this method is $O(n^2)$ where n is the number of concepts in the ontology, whereas the complexity of Algorithm 1 is $O(n^3.complexity_{scoring})$.

```
Algorithm 2. Modularization (adjacencyMatrix)
//C represents whole modularization
  C = put every node in a separate module
  for i=1 to N  // N shows size of Adjacency Matrix
    for j=1 to N
      incidenceMatrix =constructIncidenceMatrix (adjacen-
      cyMatrix);
    end for
  end for
  incidenceMatrix = Sort(incidenceMatrix);
  for i=1 to K // K shows length of adjacency matrix
    if Numberofmodules ==1
      break;
    end if
    //module(i,1) is the module for start node of edge(i)
    //module(i,2) is the module for end node of edge(i)
    c=Union(module(i,1),module(i,2));
    //if size of merged modules are more than ϵ which is
    //number of concepts/3, don't combine modules.
```

```
    if |c|>ϵ
      continue;
    else
      C=C∪{c}\{module(i,1),module(i,2)};
    end if
    //Checks the score of modularization if it is fixed
    //do not merged and terminate the algorithm.
    sc=Score(C);
    if (sc_old=sc)
      break;
    end if
    sc_old=sc;
  end for
```

4 Experimental Results

We have implemented the proposed modularization algorithms in Matlab and use Java to process the ontologies. We use F-measure [10] to evaluate our methods. F-measure is a value between zero and one and higher values show better performance. This metric compares produced modules to reference modules that are already available for tested ontologies. F-measure is computed for pairs of modules in which one module is selected from reference modularization and another is one from the generated modules. Finally, the average F-measures of all modules is computed as the F-measure of whole modularization.

$$F - measure = \frac{2 * precision * recall}{precision + recall} \qquad (7)$$

Where *precision* is the number of common concepts between two modules, divide by number of concepts in generated module. On the other hand, *recall* is calculated by dividing number of common concepts by number of reference concepts.

4.1 Dataset

We use FOAF[3], AAIR[4], BIO[5] and SWCO[6] ontologies as introduced in [10] and compared our results to concept grouping of these ontologies that are provided in their websites, and in [10]. Table 2 shows the details of these ontologies. The following provides a brief description of them:

- Friend of a Friend (FOAF) Ontology: FOAF is an ontology that describe persons, their activities and other personal information.

[3] http://xmlns.com/foaf/spec/20100101.html
[4] http://xmlns.notu.be/aair
[5] http://vocab.org/bio/0.1/.html
[6] http://data.semanticweb.org/ns/swc/ontology

- Biographical Information Ontology (BIO): BIO is an ontology to represent biographical information about people, both living and dead.
- Semantic Web Conference Ontology (SWCO): The SWCO defines concepts about academic conferences.
- Atom Activity Streams Vocabulary ontology (AAIR): This ontology is a vocabulary for describing social networking sites activities.

Table 2. Details of ontologies

Ontology	Number of modules as stated in the reference	Number of classes	Number of property
FOAF	5	13	61
BIO	5	42	33
SWCO	5	29	16
AAIR	4	41	26

In [10], they apply three algorithms: Fast Greedy Community (FGC), Walk Community (WTC) and Spin Glass Community (SGC). Because FGC and WTC are agglomerative algorithms, we compare our algorithms with them. As there are several ways introduced in [10] to represent an ontology as a graph, we choose a type of their representation of ontology in that every subject, object and predicate is represented as a separate node.

Our results are shown in Table 3. Column 1 and 2 show F-measure of our algorithms and column 4 and 5 show [10] algorithms the F-measure. The F-measure result is multiplied by 100 as it is done in [10].

Table 3. F-measure comparison of different Algorithms on different ontologies

Ontology	Alg 1	Alg 2	FGC [10]	WTC [10]
FOAF	40	30	32	34
BIO	43	33	83	79
SWCO	47	38	28	31
AAIR	50	41	51	51

Analysis of Result. The distribution of classes and properties within their concept grouping affect F-measure**.** For example for the FOAF ontology, one group just contains properties but we consider both classes and properties to modularize. In concept grouping of AAIR and SWCO, the distribution between classes and property is balanced and subclass relation is defined as the main concept. The groups of the BIO ontology contain one group for classes and four groups for properties, so our score is low. When the concept grouping contains the groups that have classes and property,

our score is good because the main objective of ontology modularization is that the modules describe subdomains.

A Simple Case Study. We also use a small ontology given in [8], which consists of six classes, and one property. In [8], they produce three modules, i.e. modules {Reference, Inproceedings, Book, Monograph}, {Author, Person}, and {hasAuthor}. However in our work, we produce two modules {Reference, Inproceedings, Book, Monograph}, and {has Author, Author, Person}. The reason for these modularizations is that while the work in [8] only considers subClassOf relations, we have considered domain/range and subClassOf relations. The modularization presented by [8] is assumed as reference, and the calculation of F-measure is shown in Table 4. F-measures comparing modules 1 and 2 are consequently 1.0 and 0.5, which are computed using equation (7). As we have only two modules, the F-measure for third module becomes zero. The average of these three gives 0.5 as the F-measure for whole modularization.

Table 4. Experimental result on sample dataset

	Module1	Module2	Module3
Precision	1.0	0.3	0
Recall	1	1	0
F-measure	1.0	0.5	0

5 Conclusion

We have proposed modularization algorithms based on semantic and structure of ontology. Semantic is considered based on assigning weight to different relationships. Furthermore, we have considered more relationships than other approaches that consider only hierarchical relation of classes. Considering more relationships from an ontology leads to making more edges in graph representation of that ontology. Thus, one can make a better decision on whether two nodes are similar.

We used neighborhood random walk distance matrix to combine semantic and structural aspects of an ontology. Each element of this matrix is calculated considering weights of almost all elements of the transition probability matrix, thus weights used in proposed method are more precise than methods, which only use weight matrix.

We have introduced a new scoring function to merge modules. The objectives of this function are to maximize the intra-module similarity and to minimize inter-module similarity. The scoring function shows how appropriately nodes have been grouped in their modules according to its objectives.

As a result, we have produced meaningful modules as we consider more relations than similar methods, and process these relations such that each edge weight has an impact on every module selection.

In future work, we plan to evaluate our experiments with other evaluation methods and other datasets to determine the efficiency of our algorithms. Furthermore, we would like to further investigate the weight of edges to improve our approach.

References

1. Parent, C., Spaccapietra, S.: An Overview of Modularity. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies. LNCS, vol. 5445, pp. 5-23. Springer, Heidelberg(2009)
2. Özacar, T., Öztürk, O., Ünalır, M.O.: ANEMONE: An Environment for Modular Ontology Development. Data & Knowledge Engineering. 70, 504-526 (2011)
3. Doran, P., Tamma, V., Payne, T.R., Palmisano, I.: An Entropy Inspired Measure for Evaluating Ontology Modularization. In: 5thInternationalConference on Knowledge Capture (KCAP'09), pp. 73-80. ACM, New York (2009)
4. Pathak, J., Johnson, T., Chute, C.: Survey of Modular Ontology Techniques and their Applications in the Biomedical Domain. Integrated Computer-Aided Engineering. 16, 225-242 (2009)
5. Zhangl, L., Liul, K., Qinl, X., Tangl, SH.: Extracting Module from OWL-DL Ontology. In: 2011International Conference on System Science, pp. 176-179.IEEE Press (2011)
6. Abadi, M.J.S, Zamanifar, K.: Producing Complete Modules in Ontology Partitioning. 2011 International Conference on Semantic Technology and Information Retrieval, pp. 137-143. IEEE Press (2011)
7. Hu, W., Zhao, Y., Qu, Y.: Partition-Based Block Matching of Large Class Hierarchies. In: Mizoguchi, R., Shi, Zh., Giunchiglia, F. (eds.) The Semantic Web – ASWC 2006. LNCS, vol. 4185, pp. 72–83. Springer, Heidelberg (2006)
8. Hu, W., Qu, Y., Cheng, G.: Matching Large Ontologies: A divide-and-conquer approach. Data & Knowledge Engineering, 67, pp. 140–160 (2008)
9. Coskun, G., Rothe, M., Teymourian, K., Paschke, A.: Applying Community Detection Algorithms on Ontologies for Identifying Concept Groups. In: Proceeding of the 5th International Workshop on Modular Ontologies (WoMO 2011), pp. 12-24. IOS Press (2011)
10. Coskun, G., Rothe, M., Paschke, A.: Ontology Content "At a Glance". In: 7th International Conference on Formal Ontology in Information Systems (FOIS 2012), pp. 147-159. IOS Press (2012)
11. Stuckenschmidt, H., Schlicht, A.: Structure-Based Partitioning of Large Ontologies. In: Stuckenschmidt, H., Parent, Ch., Spaccapietra, S. (eds.) Modular Ontologies. LNCS, vol. 5445, pp. 187-210. Springer, Heidelberg (2009)
12. Batagelj, V.: Analysis of large networks - islands. In: Dagstuhl seminar 03361: Algorithmic Aspects of Large and Complex Networks (2003)
13. Etminani, K., Rezaeian-Delui, A., Naghibzadeh, M.: Overlapped ontology partitioning based on semantic similarity measures. In: 2010 5th International Symposium on Telecommunications (IST), pp. 1013–1018. IEEE (2010)
14. Resource Description Framework (RDF), http://www.w3.org/RDF/
15. Cheng, H., Zhou, Y., Xu, Yu, J.: Clustering Large Attributed Graphs: A Balance between Structural and Attribute Similarities. ACM Transactions on Knowledge Discovery from Data. 5, 1-33 (2011)
16. Fortunato, S.: Community detection in graphs. Physics Reports. 486, 75-174 (2010)
17. Rousseeuw, P.: Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. Journal of Computational and Applied Mathematics. 20, 53–65 (1987)

Implementation and Evaluation of Forgetting In $\mathcal{ALC}$-Ontologies

Patrick Koopmann and Renate A. Schmidt

The University of Manchester, UK
{koopmanp, schmidt}@cs.man.ac.uk

Abstract. We implement and evaluate a recently introduced method to compute uniform interpolants for ontologies specified in the description logic $\mathcal{ALC}$. The aim of uniform interpolation is to reformulate an ontology such that it only uses a specified set of symbols, while preserving consequences that involve these symbols. Uniform interpolation is useful to applications in ontology engineering and modular ontologies. It is known that uniform interpolants of ontologies in $\mathcal{ALC}$ cannot always be presented in a finite way, and that their size can in the worst case be triple exponential in the size of the original ontology. These properties leave the question on how practical computing uniform interpolants is. The aim of this paper is to approach this question by implementing our recently presented method that always computes a finite representation of the uniform interpolant – either by using fixpoint logics or by extending the signature – and by undertaking an experimental evaluation of the method on a larger set of real-life ontologies.

1 Introduction

Ontologies represent information about concepts and relations (roles) using description logics, fragments of first-order logic, to allow reasoning systems to derive implicit information automatically. The signature of an ontology is the set of symbols used by the ontology. In forgetting, the aim is to remove concept or role symbols from an ontology in such a way that all logical consequences over the remaining symbols are preserved. The result of forgetting is a uniform interpolant, the original ontology restricted to a smaller signature, such that all consequences over that signature are preserved.

Uniform interpolation and forgetting have several potential applications that are interesting in the context of ontology engineering and modular ontologies. For example, an ontology to be published contains confidential parts that should not be accessible by the public. A solution to this problem is *predicate hiding* [4], which can be performed by forgetting the confidential concepts from the ontology. A related application is *ontology obfuscation* [7]. Here again, the aim is to share an ontology for re-use by other parties without giving away all of its information. Obfuscation is a technique known in the context of software engineering which transforms a program into a functionally equivalent progam that is difficult by human users to read and understand, to prevent reverse engineering.

Often, ontologies contain terms whose main function is to give structure and make the ontology accessible. By forgetting these terms, one can create an ontology whose structure is destroyed and which is not accessible by human users, while it can still be used for deriving logical entailments over the remaining concepts.

Other applications aim at analysing ontologies or ontology changes. One such application is *exhibiting hidden relations*. Often relations between different concepts are not stated explicitly but are only deducible with the help of reasoners. To get a better understanding how certain concepts relate to each other, one can compute the uniform interpolant over a signature of interest. Uniform interpolation can also be used to compute the *logical difference* between two versions of an ontology. Extending or modifying an ontology can lead to unintended results. Checking whether consequences over a specified signature are preserved in a new version can be performed by computing its uniform interpolant and testing whether it is entailed by the original ontology.

Despite these applications, there has not been much work yet to develop practical algorithms for uniform interpolation on real-life ontologies in expressive description logics. A reason for this might be that the known theoretical properties of uniform interpolation cast doubt on whether such practical methods even exist: it is known that for ontologies expressed in $\mathcal{ALC}$, uniform interpolants are not always expressible in a finite way, if $\mathcal{ALC}$ is also used to represent the uniform interpolant. Also, in the worst case, the size of the uniform interpolant can be triple exponential in the size of the original ontology [8]. These properties already hold for general ontologies expressed in $\mathcal{EL}$ [10,9]. The method presented in [7] is a first approach towards practical uniform interpolation for $\mathcal{ALC}$-ontologies, but it only ensures termination if the uniform interpolant is approximated by a given bound.

In [6], we present a method for uniform interpolation on $\mathcal{ALC}$-ontologies that always computes finite representations of uniform interpolants with the help of fixpoint operators. The target language $\mathcal{ALC}\mu$, which is $\mathcal{ALC}$ enriched with fixpoint operators, has the same complexity properties on the common reasoning tasks as $\mathcal{ALC}$ [2], but is currently not supported by most description logic reasoners. Fixpoint operators are also not supported by OWL, the standard language for representing web ontologies. The method presented in [6] gives a solution to this by simulating fixpoints using 'helper concept symbols' in the forgetting result. This way, the uniform interpolant is approximated signature-wise using a finite representation, and still preserves all consequences over the desired signature. If helper concept symbols are used in the result, the approximated interpolant is not entailed by the original ontology anymore, which limits the application of our method for computing the logical difference between ontologies. Our experimental results suggest however that this only happens for specific combinations of ontologies and signatures. For the other mentioned applications, these helper-concepts do not pose a major problem.

In order to approach the question as to whether the method is also practical for the mentioned applications, we present an experimental evaluation of the

method on real life ontologies. The results suggests that, while for some ontologies uniform interpolants are still hard to compute, there are a lot real-world cases for which the method can be used.

2 Preliminaries

Let N_c, N_r be two disjoint sets of *concept symbols* and *role symbols*. *Concepts* in $\mathcal{ALC}$ are of the following form:

$$\bot \mid \top \mid A \mid \neg C \mid C \sqcup D \mid C \sqcap D \mid \exists r.C \mid \forall r.C,$$

where $A \in N_c$, $r \in N_r$ and C and D are arbitrary concepts. $\top$, $C \sqcap D$ and $\forall r.C$ are defined as abbreviations: $\top$ stands for $\neg\bot$, $C \sqcap D$ for $\neg(\neg C \sqcup \neg D)$ and $\forall r.C$ for $\neg\exists r.\neg C$.

A TBox is a set of *axioms* of the forms $C \sqsubseteq D$ and $C \equiv D$, where C and D are concepts. $C \equiv D$ is a short-hand for the two axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. Since we are only dealing with the TBox part of an ontology, we will use the terms 'ontology' and 'TBox' interchangeably.

We write $C[A]$ to denote a concept that contains a concept symbol A, and denote the result of replacing A by a different expression E by $C[E]$. For a TBox $\mathcal{T}$, $\mathcal{T}^{[A \mapsto C]}$ denotes the result of replacing every A in $\mathcal{T}$ by C.

The semantics of $\mathcal{ALC}$ is defined as follows. An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where the *domain* $\Delta^{\mathcal{I}}$ is a nonempty set and the *interpretation function* $\cdot^{\mathcal{I}}$ assigns to each concept symbol $A \in N_c$ a subset of $\Delta^{\mathcal{I}}$ and to each role symbol $r \in N_r$ a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concepts as follows:

$$\bot^{\mathcal{I}} := \emptyset \qquad (\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}.$$

$C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is model of a TBox $\mathcal{T}$ if all axioms in $\mathcal{T}$ are true in $\mathcal{I}$. A TBox $\mathcal{T}$ is *satisfiable* if there exists a model for $\mathcal{T}$, otherwise it is *unsatisfiable*. $\mathcal{T} \models C \sqsubseteq D$ holds iff in every model of $\mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

In order to define $\mathcal{ALC}\mu$, we extend the language with a set N_v of *concept variables*. $\mathcal{ALC}\mu$ extends $\mathcal{ALC}$ with concepts of the form $\mu X.C$ and $\nu X.C$, where $X \in N_v$, and C is a concept in which X occurs as a concept symbol only positively (under an even number of negations). $\mu X.C$ is the *least fixpoint* of C on X, $\nu X.C$ the *greatest fixpoint*.

A concept variable X is *bound* if it occurs in the scope C of a fixpoint expression $\mu X.C$ or $\nu X.C$. Otherwise it is *free*. A concept is *closed* if it does not contain any free variables. Axioms in $\mathcal{ALC}\mu$ are of the form $C \sqsubseteq D$ and $C \equiv D$, where C and D are closed concepts.

Following [3], we define the semantics of fixpoint expressions. Let $\mathcal{V}$ be an *assignment function* that maps concept variables to subsets of $\Delta^{\mathcal{I}}$. $\mathcal{V}[X \mapsto W]$ denotes $\mathcal{V}$ modified by setting $\mathcal{V}(X) = W$. $C^{\mathcal{I},\mathcal{V}}$ is the interpretation of C

taking into account this assignment, and when $\mathcal{V}$ is defined for all variables in C, $C^{\mathcal{I},\mathcal{V}} = C^{\mathcal{I}}$. The semantics of fixpoint concepts is defined as follows:

$$(\mu X.C)^{\mathcal{I},\mathcal{V}} := \bigcap\{W \subseteq \Delta^{\mathcal{I}} \mid C^{\mathcal{I},\mathcal{V}[X\mapsto W]} \subseteq W\}$$
$$(\nu X.C)^{\mathcal{I},\mathcal{V}} := \bigcup\{W \subseteq \Delta^{\mathcal{I}} \mid W \subseteq C^{\mathcal{I},\mathcal{V}[X\mapsto W]}\}.$$

A *signature* Σ is a subset of $N_s \cup N_r$. $sig(E)$ denotes the concept and role symbols occurring in E, where E ranges over concept descriptions, axioms and TBoxes. Given two TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$ and a signature Σ, we say $\mathcal{T}_1$ and $\mathcal{T}_2$ are *Σ-inseparable*, in symbols $\mathcal{T}_1 \equiv_\Sigma \mathcal{T}_2$, iff for every concept inclusion α with $sig(\alpha) \subseteq \Sigma$, $\mathcal{T}_1 \models \alpha$ implies $\mathcal{T}_2 \models \alpha$ and vice versa. Given a TBox $\mathcal{T}$ and a signature Σ, $\mathcal{T}'$ is a *uniform interpolant of* $\mathcal{T}$ if $sig(\mathcal{T}') \subseteq \Sigma$ and $\mathcal{T} \equiv_\Sigma \mathcal{T}'$(Note that in contrast to conservative extensions and deductive modules no syntactical constraints are given[12]). From this definition follows that uniform interpolants for a given TBox and signature are unique modulo logical equivalence. For a given TBox and signature, we will therefore speak of *the* uniform interpolant and denote it by $\mathcal{T}^\Sigma$. Given a TBox $\mathcal{T}$ and a concept symbol A, the result of *forgetting* A *in* $\mathcal{T}$, denoted by $\mathcal{T}^{-A}$, is the uniform interpolant $\mathcal{T}^\Sigma$, where $\Sigma = sig(\mathcal{T})\backslash\{A\}$. Since $\mathcal{T}^{-A}$ entails exactly the same consequences as $\mathcal{T}$ that are not using A, it is easy to verify that $(\mathcal{T}^{-A})^{-B} \equiv (\mathcal{T}^{-B})^{-A}$. In other words forgetting a set of concept symbols one after the other always yields an equivalent TBox, regardless of the order in which symbols are processed.

3 The Method

In the following we give a brief overview of our method for computing uniform interpolants. For a more detailed description see [6]. We reduce computing of uniform interpolants to the problem of forgetting single concept symbols. In order to compute the uniform interpolant for a generic signature Σ, we forget the symbols which are not in Σ one after the other.

Given a TBox $\mathcal{T}$, the clausal form of $\mathcal{T}$, denoted by *clauses*($\mathcal{T}$), is a TBox $\mathcal{T}'$ with $\mathcal{T} \equiv_{sig(\mathcal{T})} \mathcal{T}'$, such that every axiom is of the form $\top \sqsubseteq L_0 \sqcup ... \sqcup L_n$, where every L_i is of the form A,$\neg A$, $\exists r.D$ or $\forall r.D$, with $A \in N_c$, $r \in N_r$ and $D \in N_D$. $N_D \subseteq N_c \setminus sig(\mathcal{T})$ is a set of designated concept symbols called *definer symbols*. Any TBox can be transformed into its clausal form using standard structural transformation and conjunctive normal form transformation techniques. We will refer to axioms of a clausal form TBox as *clauses* and just write $L_0 \sqcup ... \sqcup L_n$ omitting the leading $\top \sqsubseteq$. We also assume that clauses are represented as sets (that is, no disjunct occurs twice in a clause and the order of the disjuncts does not matter).

Our method to compute $\mathcal{T}^{-A}$ consists of five phases:

1. Set N=*clauses*($\mathcal{T}$).
2. Saturate N using the rules in Figure 1.
3. Filter out unnecessary clauses and group clauses of the form $\neg D \sqcup C_i$, where $D \in N_D$, into concept inclusions $D \sqsubseteq \bigsqcap C_i$.

Resolution:

$$\frac{C_1 \sqcup B \qquad C_2 \sqcup \neg B}{C_1 \sqcup C_2}$$

where B is the symbol A we want to forget or a definer symbol and provided $C_1 \sqcup C_2$ does not contain more than one negative definer literal.

Role Propagation:

$$\frac{C_1 \sqcup \forall r.D_1 \qquad C_2 \sqcup \mathsf{Q}r.D_2}{C_1 \sqcup C_2 \sqcup \mathsf{Q}r.D_3}$$

where $\mathsf{Q} \in \{\exists, \forall\}$ and D_3 is a (possibly new) definer symbol representing $D_1 \sqcap D_2$ and provided $C_1 \sqcup C_2$ does not contain more than one negative definer literal.

Fig. 1. Rules for forgetting concept symbol A

Non-cyclic definer elimination:

$$\frac{\mathcal{T} \cup \{D \sqsubseteq C\}}{\mathcal{T}^{[D \mapsto C]}} \qquad \text{provided } D \notin \mathit{sig}(C)$$

Definer purification:

$$\frac{\mathcal{T}}{\mathcal{T}^{[D \mapsto \top]}} \qquad \text{provided } D \text{ occurs only positively in } \mathcal{T}$$

Cyclic definer elimination:

$$\frac{\mathcal{T} \cup \{D \sqsubseteq C[D]\}}{\mathcal{T}^{[D \mapsto \nu X.C[X]]}} \qquad \text{provided } D \in \mathit{sig}(C[D])$$

Fig. 2. Rules for eliminating definer concept symbols

4. Apply the rules in Figure 2 exhaustively to eliminate introduced symbols.
5. Apply simplifications and represent clauses as proper concept inclusions.

The rules in Figure 1 derive all consequences based on the selected concept symbol A we want to eliminate, rendering clauses containing A superfluous for the uniform interpolant. The role propagation rule is special since it may involve the introduction of new definer symbols. Because we want to preserve the clausal form in Phase 2, in order to represent a concept conjunction $D_1 \sqcap D_2$, we introduce a new definer symbol D_3 and add two clauses $\neg D_3 \sqcup D_1$ and $\neg D_3 \sqcup D_2$ to the current clause set. In order to restrict the introduction of new definer symbols, we keep track of each introduced definer symbol and reuse them as much as possible. By doing this wisely it is possible to restrict the number of introduced definer symbols to maximally $2^{|N_D|}$.

It can be shown that if a set of clauses is saturated using the rules in Figure 1, all clauses containing the selected concept symbol A and all clauses containing

positive definer symbols that do not occur under a role restriction can be removed, and the resulting set is still Σ-inseparable with the original TBox [6]. The new definer symbols that are introduced in Phase 1 and 2 are eliminated in Phase 4 using the rules in Figure 2. These rules are motivated by Ackermann's Lemma and its generalised form, first published in [1] and [11], respectively.

If the desired target language is $\mathcal{ALC}$, the cyclic definer elimination rule cannot be applied, since it introduces fixpoint operators. In this case the cyclic definers remain in the result, which means the resulting TBox is not a uniform interpolant. It does, however, not contain A and preserves all consequences not containing A. The remaining cyclic definers can be seen as 'helper concept symbols' that help keep the result finite without using fixpoint operators. The result of applying only non-cyclic definer elimination and definer purification can be viewed as *signature-wise approximation* of the uniform interpolant. It should be noted though that the existence of cyclic definers in the returned result does not necessarily imply that there is no finite representation of the uniform interpolant in $\mathcal{ALC}$.

In [6] is it proven that our method always terminates and computes the uniform interpolant in $\mathcal{ALC}\mu$, or a signature-wise approximation.

4 Implementation

We implemented our forgetting method in Scala[1] using the OWL API.[2] Since fixpoint operators are not supported by most standards and reasoners, for practical applications it is of interest to compute only results that are expressible in $\mathcal{ALC}$. For this reason, our method does not eliminate definer symbols where this would lead to a fixpoint operator in the result. In order to make the method practical, we implemented several optimisations.

Restricting the Role Propagation Rule. Though in its presented form the calculus works correctly, in order to make the method practical, it is neccessary to apply further restrictions on the role propagation rule. The main role of the role propagation rule is to derive new clauses between which resolution on the symbol we want to forget is applicable. In order to avoid the unnecessary introduction of new clauses and definer symbols, we check beforehand whether applying role propagation contributes to any further resolution rule applications. If not, we omit its application.

Redundancy Elimination. From the proofs in [6] one can see that standard redundancy elimination techniques like tautology and subsumption deletion are compatible with our method. We also take into account subsumptions between introduced definer symbols: Note that $\neg D_1 \sqcup D_2$ implies $D_1 \sqsubseteq D_2$. With every newly introduced definer symbol we build up a subsumption hierarchy for definer symbols, which enables us to check for subsumption between literals of the forms $\exists r.D_1$ and $\forall r.D_2$. On the basis of this extended subsumption notion, we implement eager subsumption deletion and condensation as in classical

[1] `http://www.scala-lang.org`
[2] `http://owlapi.sourceforge.net`

resolution-based theorem provers. The correctness of these simplifications can be proven by adaptions of the proofs for the original method in [6].

Structural Transformation. Since the resolution rule and the role propagation rule only apply to a restricted subset of literals in the clause set, the number of clauses can be significantly reduced by using further structural transformations. For a clause C, let C^A denote the literals on which our rules apply, and $C^{\overline{A}}$ the remaining literals. We replace each set of clauses $\{C_0, ..., C_n\}$, such that $C_i^A = C_j^A$ for all $i, j < n$, by a single clause $X \sqcup C_0^A$, where X is a new concept symbol, and store the information that $X \equiv C_0^{\overline{A}} \sqcap ... \sqcap C_n^{\overline{A}}$. As soon as a clause is added to the result set, we undo this transformation and apply eager subsumption deletion on the current result set. This optimisation is influenced by the uniform interpolation method presented in [7].

Simplifications. The simplifications performed in Phase 5 are the following. Following an arbitrary ordering defined on concept symbols, we select the maximal literal of the form $\neg A$, if existent, and transform the clause into an axiom of the form $A \sqsubseteq C$. We then group all concept inclusion axioms that have the same concept A on the right hand side into a single concept inclusion. We apply several replacement rules to remove tautological or unsatisfiable subexpressions. We also detect tautological fixpoint-expressions. For any fixpoint expression $\nu X.C[X]$, if $C[\top]$ is a tautology, $\top$ is the greatest fixpoint of $C[X]$, and we can replace $\nu X.C[X]$ by $\top$. Tautological and unsatisfiable sub-expressions are detected using sound but incomplete syntactic criteria. Since the number of introduced definer symbols can be exponential in the number of role restrictions of the input ontology, it is also wise to minimise their occurrences. This is accomplished in Phase 5 by transforming disjunctions of the form $\exists r.C_0 \sqcup ... \sqcup \exists r.C_n$ into single existential role restrictions $\exists r.(C_0 \sqcup ... \sqcup C_n)$ and conjunctions of the form $\forall r.C_0 \sqcap ... \sqcap \forall r.C_n$ into single universal role restrictions $\forall r.(C_0 \sqcap ... \sqcap C_n)$.

Module extraction. To restrict the number of symbols we have to forget, we first extract the syntactic locality based $\top\bot*$-module [12] for the selected signature. This module is a subset of the original ontology that preserves all consequences over the signature, but may still contain thousands of additional symbols.

Purification. Before applying our method, we compute the negation normal form $\mathcal{T}_{NNF}$ of the input ontology $\mathcal{T}$. If a concept symbol A occurs only positively in $\mathcal{T}_{NNF}$, then $\mathcal{T}^{-A} = \mathcal{T}^{[A \mapsto \top]}$. If A occurs only negatively in $\mathcal{T}_{NNF}$, then $\mathcal{T}^{-A} = \mathcal{T}^{[A \mapsto \bot]}$. We call this transformation *purification of A*. The soundness of purification follows from the fact that in these cases the resolution rule would never be applied, what effectively means we only remove clauses containing A. Purification of A leads to an equivalent result as removing clauses containing A, but can be performed much faster. When computing uniform interpolants for our experimental evaluation, we observed that in some cases already thousands of concept symbols could be eliminated using purification.

5 Experimental Evaluation

In order to evaluate how our implementation behaves on real-life ontologies, we selected a set of ontologies from the NCBO BioPortal ontology repository.[3] The ontologies of this corpus are known to be diverse in complexity, size and structure [5]. From this corpus we selected all ontologies for which it is possible to download uncorrupted files of ontologies that could be parsed using the OWL API. We further noticed that on some ontologies, extracting $\top\bot*$-modules using the OWL API caused a runtime exception. Ontologies for which this was the case were excluded from our corpus as well.

Since our method is designed for $\mathcal{ALC}$-ontologies, we restricted the ontologies to their $\mathcal{ALC}$-fragments in the following way. Axioms that can be rewritten into $\mathcal{ALC}$ axioms in a unified way (equivalent concepts, disjoint concepts, disjoint union axioms, property range axioms and property domain axioms) were rewritten, the remaining axioms that are not in $\mathcal{ALC}$ were removed from the TBox. We further removed all ontologies where the $\mathcal{ALC}$-fragment of the TBox contained less than 5 concept symbols or consisted only of axioms of the form $A \sqsubseteq B$ and $A \equiv B$, where A and B are concept symbols. This way, we extracted a corpus of 207 ontologies for our experiments.

In these ontologies, on average 5.75% of the TBox axioms had to be removed in order to generate an $\mathcal{ALC}$-TBox, while 54 ontologies were completely expressible in $\mathcal{ALC}$.

The ontologies of the resulting corpus contain between 2 and 187,514 concept symbols (on average 5,728). The average number of axioms per ontology is 21,821.20 and the average axiom size is 4.61. The size of an axiom is defined recursively as follows: $\mathit{size}(A) = 1$, where A is a concept symbol, $\mathit{size}(\neg C) = \mathit{size}(C) + 1$, $\mathit{size}(\exists r.C) = \mathit{size}(\forall r.C) = \mathit{size}(C) + 2$, $\mathit{size}(C \sqcup D) = \mathit{size}(C \sqcap D) = \mathit{size}(C) + \mathit{size}(D) + 1$, and $\mathit{size}(C \sqsubseteq D) = \mathit{size}(C \equiv D) = \mathit{size}(C) + \mathit{size}(D) + 1$.

The experiments were run on an Intel Core i5-2400 CPU with four cores running at 3.10 GHz and 8 GB of RAM. Since our implementation does not make use of multi-threading, we ran several experiments in parallel in order to make full use of the multiple processors.

Depending on the application, it might either be interesting to forget a small set of concept symbols from the ontology (predicate hiding, ontology obfuscation, logical difference), or to restrict the ontology to a small signature (exhibit hidden relations, sharing restricted parts of an ontology). We first considered how our method performed on forgetting small sets of concept symbols. For this we selected random subsets of 5, 10, 50 and 150 concept symbols, 10 subsets in each case, from the signature of each ontology, for which we applied our method. Since the average number of concept symbols per ontology is 5,728, in most cases this represented a small subset of the overall signature. If, however, the signature of an ontology contained less than the selected number of concept symbols, we omitted the corresponding experiments. This was the case for 4, 21 and 60 ontologies for the signature sizes 10, 50 and 150, respectively.

[3] `http://bioportal.bioontology.org`

Ontologies	$\lvert sig(\mathcal{T}) \setminus \Sigma \rvert$	Timeouts	Definers Left	Average Nr. of Axioms	Average Size of Axioms	Average Duration
All	5	0.3%	1.6%	18,772.61	12.81	1.1 sec.
	10	1.0%	2.2%	19,233.38	9.21	1.1 sec.
	50	1.1%	11.3%	20,577.60	23.14	6.0 sec.
	150	3.6%	17.5%	24,627.58	60.29	18.5 sec.
NCI	50	0%	0%	138,216.99	5.37	23.5 sec.
	100	2%	0%	138,170.44	6.22	117.9 sec.
	150	3%	0%	138,127.78	6.26	121.5 sec.

Table 1. Results for forgetting small sets of concept symbols.

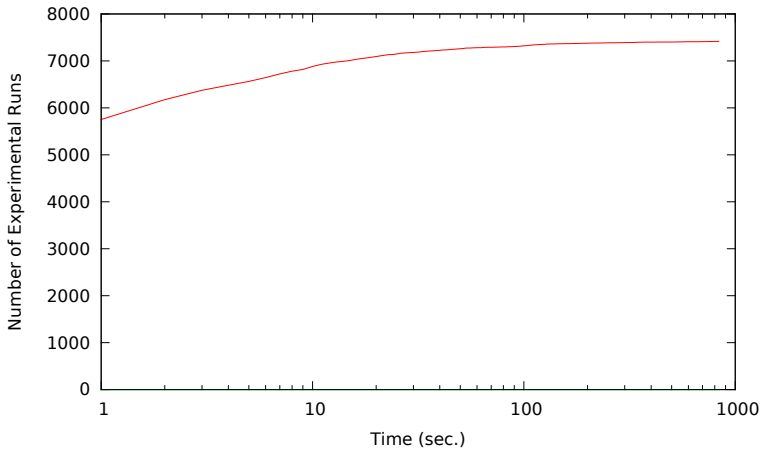


Fig. 3. Cumulative distribution of the duration of each forgetting experimental run.

We used a timeout of 1,000 seconds for each experimental run. In order to evaluate how our method performed on larger ontologies, we applied the same procedure on the $\mathcal{ALC}$-fragment of Version 13.05d of the *National Cancer Institute Thesaurus* (NCI), which was part of our corpus. The $\mathcal{ALC}$-fragment of this ontology, represented only using the operators presented in the Preliminaries Section, has 138,260 axioms of average size 5. Here, we set a higher timeout of an hour, as well as higher numbers of concept symbols, and performed 100 runs for each number.

Table 1 summarises the results of these experiments. It shows the percentage of experimental runs that could not be completed within the given time limit, the percentage of successful runs in which cyclic definer symbols remained in the results, the average number of axioms in the resulting ontology, the average size of the axioms in the result and the average duration per experimental run.

Ontologies	$\lvert\Sigma \setminus N_r\rvert$	Timeouts	Definers Left	Average Nr. of Axioms	Average Size of Axioms	Average Duration
All	5	3.5%	17.1%	3.70	627.13	5.4 sec.
	10	4.6%	20.1%	7.98	623.88	7.7 sec.
	50	8.8%	22.7%	54.84	180.48	11.8 sec.
	150	12.7%	23.1%	336.83	216.09	31.8 sec.
NCI	50	0%	15%	141.66	3,115.43	594.5 sec.
	100	4%	12%	335.28	1,876.51	927.0 sec.
	150	7%	15%	568.69	1,751.58	1,389.2 sec.

Table 2. Results for computing uniform interpolants over small signatures.

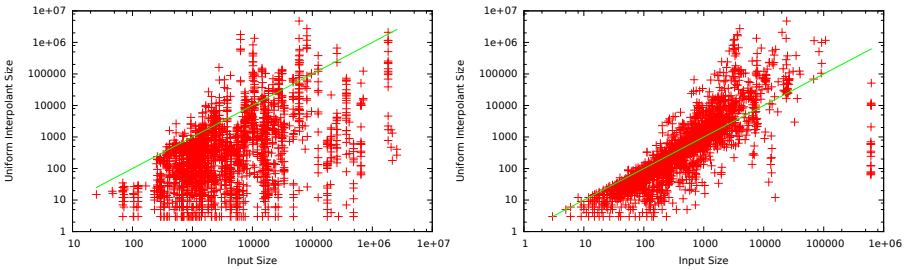


Fig. 4. Sizes of extracted modules and corresponding uniform interpolants.

The size of the ontologies remained mostly unchanged, which was due to the fact that a major part of the ontology was not touched by the method if the concept symbols were only used in a small subset.

With an increasing number of forgotten concept symbols the number of cases in which definer symbols are left in the result rose slightly, but in 93% of the cases the result could be represented finitely without definer symbols. In 99% of the cases our method was able to compute the forgetting result in the set time limit. The average duration suggests that a much smaller timeout could already have led to similar results. Figure 3 shows the cumulative distribution of the durations of each experimental run. It shows that nearly 6,000 out of 7,426 experimental runs could be performed within less than one second, which suggests that for most cases, forgetting small sets of concepts is actually a cheap operation.

Next we wanted to evaluate how good our method performed on restricting the signature of an ontology to a small set of concept symbols. Since computing uniform interpolants for small signatures is much more computationally expensive as forgetting small sets of concept symbols, we performed the experiments only on a subset of the original corpus, for which we randomly selected 170 ontologies, and performed 5 experimental runs for each ontology and sample size. The results are summarised in Table 2.

The effect of uniform interpolation was more apparent in these cases. In 20.1% of the cases, the computed uniform interpolant would have used fixpoint

operators. Even if only 5 concept symbols were used in the result, the average axiom size was 627. In case of the NCI ontology, the average size of an axiom was even higher. The main reason for this is that much more information about the role structure of the ontology and disjointnesses between concepts had to be represented in fewer axioms.

Figure 4 plots the sizes of input ontologies and the sizes of the extracted modules against the sizes of the signature-wise approximated uniform interpolants. Interestingly, in most cases the computed uniform interpolant was of similar size or smaller than the corresponding module. In 90.0% of the cases, the resulting ontology was smaller than the input ontology, and in 75.9% of the cases, it was smaller than the corresponding $\top\bot*$-module. In the most extreme case the uniform interpolant was however 559 times bigger than the corresponding $\top\bot*$-module.

The performance on our method strongly depended on how distributed the concept symbols to be forgotten are in the ontology, and how many additional symbols remained in the module. The biggest effect on computation time and output size was caused if the concept symbols to be forgotten occurred in high numbers nested under role restrictions, since the role propagation rule had to be applied more often in these cases. This lead to a high number of clauses and seemed to be the main cause for timeouts.

The corpora used for the experiments, as well as the implementation, can be found under `http://www.cs.man.ac.uk/~koopmanp/womo_experiments`.

6 Conclusion

We implemented and evaluated a recently presented method to compute uniform interpolants of $\mathcal{ALC}$-ontologies. Uniform interpolation has a lot of potential applications in ontology engineering and modular ontologies. It is known that uniform interpolants of $\mathcal{ALC}$-ontologies cannot always be represented in a finite way in $\mathcal{ALC}$, and their size is in the worst case triple exponential in the size of the input ontology. We evaluated an implementation of uniform interpolation to investigate how these theoretical properties affect uniform interpolation of $\mathcal{ALC}$-fragments of real-life ontologies. Our method computes uniform interpolants for $\mathcal{ALC}\mu$, which is $\mathcal{ALC}$ extended with fixpoint operators, to enable the finite representation of uniform interpolants in all cases. Since fixpoint operators are not supported by most standards and reasoners, our implementation uses helper concept symbols in the result, which means the computed ontologies approximate the uniform interpolant signature-wise. Our experiments showed however, that in a majority of cases this was not needed, since the uniform interpolant could be represented without fixpoint operators. Our experiments suggest that, even though the worst case complexity of the size of uniform interpolants is triple exponential, in reality, the situation where the interpolant is exponential rarely occurs. In fact, in most cases uniform interpolants could be computed in a few seconds, and were even smaller than the input ontologies. These results suggest

that, even though computing uniform interpolation for complex ontologies can be expensive, there are a lot of applications where it is practical.

In contrast to the earlier approaches on uniform interpolation of $\mathcal{ALC}$-ontologies presented [13,8], our method proceeds in a focused way in the sense that only derivations on the currently selected symbol to be forgotten are computed. This enables our method to perform efficiently on larger ontologies, but a trade-off is that our method will not always compute an interpolant in $\mathcal{ALC}$ without fixpoint operators if it exists. To illustrate the problem, consider the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A \sqcup B, B \sqsubseteq \exists r.B\}$. When forgetting B, our method computes the TBox $\mathcal{T}^{-B} = \{A \sqsubseteq \exists r.A \sqcup \nu X.\exists r.X\}$, since it only considers derivations on B. The fixpoint expression in this ontology is however redundant, since $A \sqsubseteq \exists r.A$ already entails all consequences of the form $A \sqsubseteq \exists r^n.\top$. Note that while in this example the redundancy is quite obvious, in general it will be more hidden. In future it would be desirable to find an efficient way to deal with these kind of situations.

References

1. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Mathematische Annalen 110(1), 390–413 (1935)
2. Bradfield, J., Stirling, C.: Modal mu-calculi. In: Handbook of Modal Logic, Studies in Logic and Practical Reasoning, vol. 3, pp. 721–756. Elsevier (2007)
3. Calvanese, D., Giacomo, G.D., Lenzerini, M.: Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Proc. IJCAI '99. pp. 84–89. Morgan Kaufmann (1999)
4. Grau, B.C., Motik, B.: Reasoning over ontologies with hidden content: The import-by-query approach. J. of Artificial Intelligence Research 45, 197–255 (2012)
5. Horridge, M., Parsia, B., Sattler, U.: The state of bio-medical ontologies. Bio-Ontologies 2011 (2011)
6. Koopmann, P., Schmidt, R.A.: Uniform Interpolation of $\mathcal{ALC}$-Ontologies Using Fixpoints. In: Proc. FroCoS'13. Springer (2013), to appear.
7. Ludwig, M., Konev, B.: Towards Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes. http://lat.inf.tu-dresden.de/research/papers/2013/LuKo-DL-2013.pdf
8. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proc. IJCAI '11. pp. 989–995. AAAI Press (2011)
9. Nikitina, N.: Forgetting in General $\mathcal{EL}$ Terminologies. Proc. DL '11, CEUR-WS.org (2011)
10. Nikitina, N., Rudolph, S.: ExpExpExplosion: Uniform interpolation in general $\mathcal{EL}$ terminologies. In: Proc. ECAI'12. pp. 618–623. IOS Press (2012)
11. Nonnengart, A., Szałas, A.: A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In: Logic at Work, pp. 307–328. Springer (1999)
12. Sattler, U., Schneider, T., Zakharyaschev, M.: Which Kind of Module Should I Extract? In: Proc. DL'09. CEUR-WS.org (2009)
13. Wang, Z., Wang, K., Topor, R., Zhang, X.: Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In: Proc. ECAI '10. pp. 47–52. IOS Press (2010)

Module Extraction for Acyclic Ontologies

William Gatens, Boris Konev, and Frank Wolter

The University of Liverpool, UK

Abstract. We present an implementation (AMEX) of a module extraction algorithm for acyclic description logic ontologies. The implementation uses a QBF solver (sKizzo) to check whether one ontology is a conservative extension of another ontology relativised to interpretations of cardinality one. We evaluate AMEX by applying it to NCI (the National Cancer Institute Thesaurus) and by comparing the extracted AMEX-modules with locality-based modules. We also present experiments for a hybrid approach in which AMEX and locality-based module extraction are applied iteratively to NCI.

1 Introduction

Module extraction is the task of computing, given an ontology and a signature Σ of interest, a subset (called module) of the ontology such that for certain applications that use the signature Σ only, the original ontology can be equivalently replaced by the module [14]. In most applications of module extraction it is desirable to compute a small (and, if possible, even minimal) module. In logic-based approaches to module extraction, the most robust and popular way to define modules is via model-theoretic Σ-inseparability, where two ontologies are called Σ-inseparable iff the Σ-reducts of their models coincide. Then, a Σ-module of an ontology is defined as a Σ-inseparable subset of the ontology [10, 7, 3, 8]. It is often helpful and necessary to refine this notion of Σ-module by considering self-contained Σ-modules (modules that are inseparable from the ontology not only w.r.t. Σ but also w.r.t. their own signature) and depleting modules (modules such that the remaining axioms in the ontology say nothing about Σ and the signature of the module, that is, these remaining axioms are inseparable from the empty ontology w.r.t. Σ and the signature of the module). Note that every depleting module is a self-contained module is a module. In all three cases it is often not possible to compute Σ-modules: by results in [8, 11], for acyclic $\mathcal{ALC}$-TBoxes and general $\mathcal{EL}$-TBoxes it is undecidable whether a given subset of a TBox is a (self-contained, depleting) Σ-module. The "maximal" description logics (DLs) for which efficient algorithms computing minimal self-contained and depleting Σ-modules have been developed are acyclic $\mathcal{EL}$ [8] and DL-Lite [9, 10, 6].[1] For this reason, for module extraction for ontologies given in expressive DLs or other expressive ontology languages one has to employ approximation algorithms: instead of computing a minimal (self-contained, depleting) Σ-module, one computes some (self-contained, depleting) Σ-module and

[1] For typical DL-Lite dialects, model-theoretic Σ-inseparability is decidable. Experimental evaluations of module extraction algorithms are, however, available only for language dependent notions of inseparability.

the main research problem is to minimise the size of the module (or, equivalently, to approximate minimal modules). Currently, the most popular and successful approximation algorithm is based on locality and computes so-called $\top\bot^*$-modules [4]. The size of $\top\bot^*$-modules and the performance of algorithms extracting $\top\bot^*$-modules has been analysed systematically and in great detail [4]. However, since no alternative logically sound and implemented module extraction algorithms are available for expressive DLs, it remained open how large and significant the difference between $\top\bot^*$-modules and minimal modules is and in how far it is possible to improve upon the approximation obtained by $\top\bot^*$-modules.[2]

The contribution of this paper is as follows.

1. We extend the module extraction algorithm introduced in [8] from acyclic $\mathcal{ALCI}$-TBoxes to acyclic $\mathcal{ALCQI}$-TBoxes with repeated concept inclusions and present a number of optimisations of the algorithm given in [8]. We note that our extraction algorithm is polynomial time except that it uses a QBF-solver as an oracle.
2. We describe our implementation, called AMEX, of this module extraction algorithm. AMEX is available from `http://www.csc.liv.ac.uk/~wgatens/software/amex.html`.
3. We evaluate its efficiency in experiments with NCI and compare the size of the computed AMEX-modules with the size of $\top\bot^*$-modules.
4. We introduce a hybrid approach to module extraction in which $\top\bot^*$-module extraction and AMEX-module extraction are applied iteratively. Unlike AMEX on its own, this hybrid approach is applicable to arbitrary description logic TBoxes. We demonstrate that on some inputs both AMEX and the hybrid approach lead to significant reductions in the size of modules.

2 Preliminaries

We use standard notation from logic and description logic (DL), details can be found in [1]. In a DL, concepts are constructed from countably infinite sets $\mathsf{N_C}$ of *concept names* and $\mathsf{N_R}$ of *role names* using the concept constructors defined by the DL. For example, *$\mathcal{ALCQI}$-concepts* are built according to the rule

$$C ::= A \quad | \quad \top \quad | \quad \neg C \quad | \quad \geq n\, r.C \quad | \quad \geq n\, r^{-}.C \quad | \quad C \sqcap D,$$

where $A \in \mathsf{N_C}$, n is a natural number, and $r \in \mathsf{N_R}$. As usual, we use the following abbreviations: $\bot$ denotes $\neg\top$, $\exists r.C$ denotes $\geq 1\ r.C$, $\forall r.C$ denotes $\neg\exists\ r.\neg C$, $C \sqcup D$ denotes $\neg(\neg C \sqcap \neg D)$, $\leq n\ r.C$ denotes $\neg(\geq (n+1)\ r.C)$, and $(= n\ r.C)$ for $((\geq n\ r.C) \sqcap (\leq n\ r.C))$.

A *general TBox* $\mathcal{T}$ is a finite set of *axioms*, where an axiom can be either a *concept inclusion (CI)* $C \sqsubseteq D$ or a *concept equality (CE)* $C \equiv D$, where C and D are concepts. A general TBox $\mathcal{T}$ is *acyclic* if all its axioms are of the form $A \sqsubseteq C$ or $A \equiv C$, where

[2] An implementation of semantic locality-based $\Delta\emptyset^*$-modules and a comparison between $\top\bot^*$ and $\Delta\emptyset^*$-modules have been presented in [4]; however, the authors found no significant difference between the two approaches. A promising approach to refine $\top\bot^*$-module extraction has recently been presented in [12], but an implemented system is not yet publicly available.

$A \in \mathsf{N_C}$, no concept name occurs more than once on the left-hand side and $A \not\prec_{\mathcal{T}}^{+} A$, for any $A \in \mathsf{N_C}$, where $\prec_{\mathcal{T}}^{+}$ is the transitive closure of the relation $\prec_{\mathcal{T}} \subseteq \mathsf{N_C} \times (\mathsf{N_C} \cup \mathsf{N_R})$ defined by setting $A \prec_{\mathcal{T}} X$ iff there exists an axiom of the form $A \sqsubseteq C$ or $A \equiv C$ in $\mathcal{T}$ with $X \in \mathsf{sig}(C)$.

The semantics of DLs is given by *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the *domain* $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is an *interpretation function* that maps each $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is inductively expanded to complex concepts C in the standard way [1]. An interpretation $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$ (written $\mathcal{I} \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, it *satisfies* a CE $C \equiv D$ (written $\mathcal{I} \models C \equiv D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$.

3 Module Extraction

In this section we define depleting modules and give an algorithm computing depleting modules of acyclic $\mathcal{ALCQI}$-TBoxes using a QBF solver. To cover the NCI Thesaurus, we also extend our extraction algorithm to TBoxes that are acyclic except that they contain repeated concept inclusions. The results presented in this section are extensions of the results presented in [8] for acyclic $\mathcal{ALCI}$-TBoxes to acyclic $\mathcal{ALCQI}$-TBoxes with repeated concept inclusions.

A *signature* Σ is a finite subset of $\mathsf{N_C} \cup \mathsf{N_R}$. The signature $\mathsf{sig}(C)$ ($\mathsf{sig}(\alpha)$, $\mathsf{sig}(\mathcal{T})$) of a concept C (axiom α, TBox $\mathcal{T}$, resp.) is the set of concept and role names that occur in C ($\alpha, \mathcal{T}$, resp.). If a $\mathsf{sig}(C) \subseteq \Sigma$ we call C a Σ-concept. The *Σ-reduct* $\mathcal{I}|_\Sigma$ of an interpretation $\mathcal{I}$ is obtained from $\mathcal{I}$ by setting $\Delta^{\mathcal{I}|_\Sigma} = \Delta^{\mathcal{I}}$, and $X^{\mathcal{I}|_\Sigma} = X^{\mathcal{I}}$ for all $X \in \Sigma$, and $X^{\mathcal{I}|_\Sigma} = \emptyset$ for all $X \notin \Sigma$. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes and Σ a signature. Then $\mathcal{T}_1$ and $\mathcal{T}_2$ are *Σ-inseparable*, in symbols $\mathcal{T}_1 \equiv_\Sigma \mathcal{T}_2$, if

$$\{\mathcal{I}|_\Sigma \mid \mathcal{I} \models \mathcal{T}_1\} \;=\; \{\mathcal{I}|_\Sigma \mid \mathcal{I} \models \mathcal{T}_2\}.$$

It is proved in [8] that TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are Σ-inseparable if, and only if, $\mathcal{T}_1 \models \varphi$ iff $\mathcal{T}_2 \models \varphi$ holds for any second-order sentence φ using symbols for Σ only. Thus, Σ-inseparable TBoxes cannot be distinguished by their second-order consequences formulated in Σ. We use Σ-inseparability to define modules.

Definition 1. *Let $\mathcal{M} \subseteq \mathcal{T}$ be TBoxes and Σ a signature. Then $\mathcal{M}$ is a* depleting Σ-module *of $\mathcal{T}$ if $\mathcal{T} \setminus \mathcal{M} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$.*

Every depleting module $\mathcal{M}$ of $\mathcal{T}$ is inseparable from the $\mathcal{T}$ for its signature [8], that is, if $\mathcal{M}$ is a depleting Σ-module of $\mathcal{T}$ then $\mathcal{T} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \mathcal{M}$, and, in particular, $\mathcal{T} \equiv_\Sigma \mathcal{M}$. Thus, a TBox and its depleting Σ-module can be equivalently replaced by each other in applications which concern Σ only. Unfortunately, checking if a subset $\mathcal{M}$ of $\mathcal{T}$ is a depleting Σ-module of $\mathcal{T}$ for some given signature Σ is undecidable already for general TBoxes formulated in $\mathcal{EL}$ and for acyclic $\mathcal{ALC}$-TBoxes [8, 11].

We therefore consider syntactic restrictions that ensure that depleting modules become decidable. We say that an acyclic TBox $\mathcal{T}$ *has a direct Σ-dependency*, for some signature Σ, if there exists $\{A, X\} \subseteq \Sigma$ with $A \prec_{\mathcal{T}}^{+} X$; otherwise we say that $\mathcal{T}$ *has no direct Σ-dependencies*. Although one can construct TBoxes $\mathcal{T}$ and depleting Σ-modules $\mathcal{M}$ of $\mathcal{T}$ such that $\mathcal{T} \setminus \mathcal{M}$ contains direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies (see

[8]), for typical depleting Σ-modules $\mathcal{M}$, the set $\mathcal{T} \setminus \mathcal{M}$ should not contain direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies because such dependencies indicate a semantic link between two distinct symbols in $\Sigma \cup \mathsf{sig}(\mathcal{M})$. The main advantage of making the assumption that $\mathcal{T} \setminus \mathcal{M}$ has no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies is that it becomes decidable whether $\mathcal{T} \setminus \mathcal{M} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$ [8]. The following lemma directly implies this decidability result. For an acyclic TBox $\mathcal{T}$ and a signature Σ let

$$\mathsf{Lhs}_\Sigma(\mathcal{T}) = \{A \bowtie C \in \mathcal{T} \mid A \in \Sigma \text{ or } \exists X \in \Sigma\ (X \prec_\mathcal{T}^+ A)\}.$$

The following is proved in [8] for acyclic $\mathcal{ALCI}$-TBoxes. The generalization to $\mathcal{ALCQI}$ is straightforward and omitted.

Lemma 1. *Let $\mathcal{T}$ be an acyclic $\mathcal{ALCQI}$-TBox. If $\mathcal{T} \setminus \mathcal{M}$ has no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies then the following conditions are equivalent for every $\mathcal{W} \subseteq \mathcal{T} \setminus \mathcal{M}$:*

(a) $\mathcal{W} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$;
(b) for every $\mathcal{I}$ with $|\Delta^\mathcal{I}| = 1$ there exists a model $\mathcal{J}$ of $\mathsf{Lhs}_{\Sigma \cup \mathsf{sig}(\mathcal{M})}(\mathcal{W})$ such that $\mathcal{I}|_{\Sigma \cup \mathsf{sig}(\mathcal{M})} = \mathcal{J}|_{\Sigma \cup \mathsf{sig}(\mathcal{M})}$.

Since the condition (b) of Lemma 1 refers to interpretations with a singleton domain, it can be checked by reduction to validity of a quantified Boolean formula: take a propositional variable p_A for each name $A \in \Sigma \cup \mathsf{sig}(\mathcal{M})$ and a (distinct) propositional variable q_X for each symbol $X \in \mathsf{sig}(\mathcal{T}) \setminus (\Sigma \cup \mathsf{sig}(\mathcal{M}))$. Translate concepts D in the signature $\mathsf{sig}(\mathcal{T})$ into propositional formulas $D^\dagger$ by setting

$$\begin{array}{rll}
A^\dagger &= p_A & \text{for all } A \in \Sigma \cup \mathsf{sig}(\mathcal{M}) \\
A^\dagger &= q_A & \text{for all } A \in \mathsf{sig}(\mathcal{T}) \setminus (\Sigma \cup \mathsf{sig}(\mathcal{M})) \\
(D_1 \sqcap D_2)^\dagger &= D_1^\dagger \wedge D_2^\dagger & \\
(\neg D)^\dagger &= \neg D^\dagger & \\
(\geq 1\ r.D)^\dagger = (\geq 1\ r^-.D)^\dagger &= q_r \wedge D^\dagger & \text{for all } r \in \mathsf{sig}(\mathcal{T}) \\
(\geq n\ r.D)^\dagger = (\geq n\ r^-.D)^\dagger &= \bot & \text{for all } n > 1 \text{ and } r \in \mathsf{sig}(\mathcal{T})
\end{array}$$

Now let

$$\mathcal{T}^\dagger = \bigwedge_{C \sqsubseteq D \in \mathcal{T} \setminus \mathcal{M}} C^\dagger \to D^\dagger \wedge \bigwedge_{C \equiv D \in \mathcal{T} \setminus \mathcal{M}} C^\dagger \leftrightarrow D^\dagger$$

and let $\boldsymbol{p}$ denote the sequence of variables p_A, $A \in \Sigma \cup \mathsf{sig}(\mathcal{M})$, and $\boldsymbol{q}$ denote the sequence of variables q_X, $X \in \mathsf{sig}(\mathcal{T}) \setminus (\Sigma \cup \mathsf{sig}(\mathcal{M}))$. One can show that condition (b) of Lemma 1 holds if, and only if, the QBF $\varphi_\mathcal{T} := \forall \boldsymbol{p} \exists \boldsymbol{q} \mathcal{T}^\dagger$ is valid. Thus, for TBoxes with no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies the separability check can be implemented using a QBF solver.

Lemma 1 can be used directly for a naïve module extraction algorithm which goes through all subsets of $\mathcal{T}$ to identify a smallest possible $\mathcal{M}$ such that $\mathcal{T} \setminus \mathcal{M}$ has no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies and $\mathcal{T} \setminus \mathcal{M} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$. Instead, we consider a refined goal-oriented approched based on the notion of a *separability causing axiom*. Let $\mathcal{M} \subseteq \mathcal{T}$ and a signature Σ be such that $\mathcal{T} \setminus \mathcal{M}$ has no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies. We call an axiom $A \bowtie C \in \mathcal{T} \setminus \mathcal{M}$, where $\bowtie \in \{\sqsubseteq, \equiv\}$, separability causing if there exists a $\mathcal{W} \subseteq \mathcal{T} \setminus \mathcal{M}$ such that

$$A \bowtie C \in \mathcal{W}; \qquad (\mathcal{W} \setminus \{A \bowtie C\}) \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset; \qquad \mathcal{W} \not\equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset.$$

Input: Acyclic $\mathcal{ALCQI}$ TBox $\mathcal{T}$, Signature Σ
Apply Rules 1 and 2 exhaustively, preferring Rule 1.
Output: (Minimal) Module $\mathcal{M}$ s.t $\mathcal{T} \setminus \mathcal{M} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$ and $\mathcal{T} \setminus \mathcal{M}$ has no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$ dependencies.

(R1) If an axiom $A \bowtie C \in \mathcal{T} \setminus \mathcal{M}$ is such that $A \in \Sigma \cup \mathsf{sig}(\mathcal{M}))$ and $A \prec^{+}_{\mathcal{T} \setminus \mathcal{M}} X$, for some $X \in (\Sigma \cup \mathsf{sig}(\mathcal{M}))$, then set $\mathcal{M} := \mathcal{M} \cup \{A \bowtie C\}$

(R2) If an axiom $A \bowtie C \in \mathcal{T} \setminus \mathcal{M}$ is a *separability causing axiom* then set $\mathcal{M} := \mathcal{M} \cup \{A \bowtie C\}$

Fig. 1. Module extraction in $\mathcal{ALCQI}$

Input: TBox $\mathcal{T}$, subset $\mathcal{M} \in \mathcal{T}$ and signature Σ such that
$\mathcal{T} \setminus \mathcal{M}$ contains no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies and $\mathcal{T} \setminus \mathcal{M} \not\equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$
Output: Separability causing axiom α

$\mathcal{W} = \mathsf{lastAdded} = \mathit{topHalf}(\mathsf{Lhs}_{\Sigma \cup \mathsf{sig}(\mathcal{M})}(\mathcal{T} \setminus \mathcal{M}))$
$\mathsf{lastRemoved} = \mathit{bottomHalf}(\mathsf{Lhs}_{\Sigma \cup \mathsf{sig}(\mathcal{M})}(\mathcal{T} \setminus \mathcal{M}))$
while $\mathsf{lastAdded} \neq \emptyset$ **do**
    if $\mathcal{W} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$ **then**
        $\mathsf{lastAdded} = \mathit{topHalf}(\mathsf{lastRemoved})$
        $\mathcal{W} = \mathcal{W} \cup \mathsf{lastAdded}$
        $\mathsf{lastRemoved} = \mathsf{lastRemoved} \setminus \mathsf{lastAdded}$
    else
        $\mathsf{lastRemoved} = \mathit{bottomHalf}(\mathsf{lastAdded})$
        $\mathcal{W} = \mathcal{W} \setminus \mathsf{lastRemoved}$
        $\mathsf{lastAdded} = \mathsf{lastAdded} \setminus \mathsf{lastRemoved}$
return the last axiom of $\mathcal{W}$

Fig. 2. Finding separability causing axiom

Clearly, if $\mathcal{T} \setminus \mathcal{M} \not\equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$ then $\mathcal{T} \setminus \mathcal{M}$ contains a separability causing axiom.

The algorithm computing a depleting Σ-module of acyclic $\mathcal{ALCQI}$-TBoxes is now given in Figure 1. In the algorithm, the extraction of depleting Σ-modules is broken into the rules **R1** and **R2**. The rule **R1** checks for direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies. The rule **R2** implements an inseparability check. Notice that **R2** only applies when **R1** is not applicable, that is only if $\mathcal{T} \setminus \mathcal{M}$ contains no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies. Notice that applications of the **R1** rule can lead to axioms unnecessarily being included into the module; but such is the price we pay for regaining the decidability of the inseparability check.

To reduce the number of calls to the QBF solver, rule **R2** is implemented as binary search. We first consider $\mathcal{T} \setminus \mathcal{M}$ itself as $\mathcal{W}$. If $\mathcal{T} \setminus \mathcal{M} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$ then $\mathcal{T} \setminus \mathcal{M}$ contains no separability causing axioms. Otherwise, we consider $\mathcal{W}$ to be equal to the top half of $\mathcal{T} \setminus \mathcal{M}$ (we treat $\mathcal{T} \setminus \mathcal{M}$ as an ordered set). We then check if $\mathcal{W} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$ and, if this is the case, we grow $\mathcal{W}$ from the bottom and if not, we half it again as shown in

Figure 2. In the worst case we perform $\log_2(|\mathcal{T} \setminus \mathcal{M}|)$ inseparability checks to locate a separability causing axiom.

To summarise, the module extraction algorithm in Figure 1 runs in polynomial time with each call to the QBF solver being treated as a constant time oracle call. Note that QBF solvers have been used before in module extraction [9, 10], but the task solved by the solver here is completely different from its task in [9, 10].

It should be clear that if neither **R1** nor **R2** is applicable then $\mathcal{T} \setminus \mathcal{M} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$ and so the output of the algorithm in Figure 1 is a depleting Σ-module. By a straightforward generalisation of the results of [8] to $\mathcal{ALCQI}$ one can actually show that the module computed in Figure 1 is uniquely determined:

Theorem 1. *Given an acyclic $\mathcal{ALCQI}$ TBox $\mathcal{T}$ and signature Σ the algorithm in Figure 1 computes the unique minimal depleting Σ-module s.t. $\mathcal{T} \setminus \mathcal{M}$ contains no direct $\Sigma \cup \mathsf{sig}(\mathcal{M})$-dependencies.*

Note that the minimality condition in the theorem means that for any $\mathcal{M}' \subseteq \mathcal{T}$ such that $\mathcal{T} \setminus \mathcal{M}'$ has no direct $\Sigma \cup \mathsf{sig}(\mathcal{M}')$-dependencies and $\mathcal{T} \setminus \mathcal{M}' \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M}')} \emptyset$ we have $\mathcal{M} \subseteq \mathcal{M}'$. It is, however, still possible that there exists a $\mathcal{M}'' \subseteq \mathcal{T}$ with $\mathcal{T} \setminus \mathcal{M}'' \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M}'')} \emptyset$, $\mathcal{M} \not\subseteq \mathcal{M}''$ and such that $\mathcal{T} \setminus \mathcal{M}''$ has some direct $\Sigma \cup \mathsf{sig}(\mathcal{M}'')$-dependencies.

Example 1. We apply the algorithm in Figure 1 to the following acyclic TBox $\mathcal{T}$ inspired by the NCI Thesaurus (we have simplified some axioms and abbreviated 'kidney' with K, 'ureter' with U and 'tract' with T)

$$\mathsf{Renal\_Pelvis\_and\_U} \sqsubseteq \exists \mathsf{partOf}.\mathsf{K\_and\_U} \quad (1)$$
$$\mathsf{K\_and\_U\_Neoplasm} \equiv \mathsf{U\_T\_Neoplasm} \sqcap (\forall \mathsf{hasSite}.\mathsf{K\_and\_U}) \quad (2)$$
$$\mathsf{Malignt\_U\_T\_Neoplasm} \equiv \mathsf{U\_T\_Neoplasm} \sqcap (\forall \mathsf{hasAbnCell}.\mathsf{Malignt\_Cell}) \quad (3)$$
$$\mathsf{Benign\_U\_T\_Neoplasm} \equiv \mathsf{U\_T\_Neoplasm} \sqcap (\forall \mathsf{excludesAbnCell}.\mathsf{Malignt\_Cell}) \quad (4)$$

and $\Sigma = \{\mathsf{Malignt\_U\_T\_Neoplasm}, \mathsf{K\_and\_U\_Neoplasm}, \mathsf{Renal\_Pelvis\_and\_U}\}$. It can be seen that **R1** is not applicable. To see why $\mathsf{Lhs}_\Sigma(\mathcal{T}) \not\equiv_\Sigma \emptyset$ consider an interpretation $\mathcal{I}$ with $\Delta^\mathcal{I} = \{d\}$ such that $\mathsf{Renal\_Pelvis\_and\_U}^\mathcal{I} = \mathsf{Malignt\_U\_T\_Neoplasm}^\mathcal{I} = \{d\}$ and $\mathsf{K\_and\_U\_Neoplasm}^\mathcal{I} = \emptyset$. It can be readily checked for any $\mathcal{J}$ with $\mathcal{J}|_\Sigma = \mathcal{I}|_\Sigma$ that $\mathcal{J} \not\models \mathcal{T}$. This check can be delegated to a QBF solver as explained above.

The algorithm in Figure 2 splits $\mathsf{Lhs}_\Sigma(\mathcal{T})$ into two parts, $\mathsf{lastAdded} = \{(1), (2)\}$ and $\mathsf{lastRemoved} = \{(3)\}$. For $\mathcal{W} = \mathsf{lastAdded}$ it can be checked that $\mathcal{W} \equiv_\Sigma \emptyset$. Then the algorithm grows $\mathcal{W}$ with (the upper part of) lastRemoved. The same argument as above shows that for $\mathcal{W} = \{(1), (2), (3)\}$ we have $\mathcal{W} \not\equiv_\Sigma \emptyset$ and so the algorithm identifies (3) as a separability causing axiom. After applying the rule **R2**, $\Sigma \cup \mathsf{sig}(\mathcal{M}) = \{\mathsf{Malignt\_U\_T\_Neoplasm}, \mathsf{K\_and\_U\_Neoplasm}, \mathsf{Renal\_Pelvis\_and\_U}, \mathsf{U\_T\_Neoplasm}, \mathsf{hasAbnCell}\}$ and then the rule **R1** adds axioms (1) and (2) to $\mathcal{M}$.

It can be seen that neither **R1** nor **R2** applies to $\mathcal{T} \setminus \mathcal{M} = \{(4)\}$ and the computation concludes with $\mathcal{M} = \{(1), (2), (3)\}$. Notice that although $\{(4)\} \equiv_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \emptyset$, axiom (4) is neither Δ- nor $\emptyset$-local for $\Sigma \cup \mathsf{sig}(\mathcal{M})$ and so the $\top\bot^*$-module of $\mathcal{T}$ w.r.t. Σ coincides with $\mathcal{T}$ (see below and [3] for definitions).

It is often the case (e.g., for the NCI Thesaurus) that a real-world ontology satisfies all conditions for acyclic TBoxes with the exception that it contains multiple concept inclusions of the form $A \sqsubseteq C_1, \ldots, A \sqsubseteq C_n$. We call such TBoxes *acyclic with repeated*

concept inclusions. Clearly, one can convert such a TBox into an equivalent acyclic TBox by replacing all repeated concept inclusions of the form $A \sqsubseteq C_1$,..., $A \sqsubseteq C_n$ with $A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$. However, such an explicit conversion is an unattractive solution for module extraction because if such an axiom is added to a Σ-module the signature of the module now contains every symbol in the definition of every repeated name increasing the size of the resulting module considerably. The approach we take to handle acyclic TBoxes with repeated concept inclusions is to introduce fresh concept names for different repeated occurrences of a concept name in the left-hand side of concept inclusions, extract modules from the resulting acyclic TBox and then substitute away the added names as follows.

Theorem 2. *Let $\mathcal{T}$ be an acyclic TBox with repeated concept inclusions and Σ a signature. Let $\mathcal{T}'$ consist of all $A \bowtie C \in \mathcal{T}$ which are not repeated in $\mathcal{T}$ and all $A'_1 \sqsubseteq C_1, \ldots, A'_n \sqsubseteq C_n, A \sqsubseteq A'_1 \sqcap \ldots \sqcap A'_n$, where $A \sqsubseteq C_1$,..., $A \sqsubseteq C_n$ are all concept inclusions in $\mathcal{T}$ with A on the left hand side, $n > 1$, and $A'_1, \ldots, A'_n$ are fresh concept names.*

Let $\mathcal{M}'$ be a depleting Σ-module of $\mathcal{T}'$ and let $\mathcal{M}$ be obtained from $\mathcal{M}'$ by dropping the added axioms of the form $A \sqsubseteq A'_1 \sqcap \ldots \sqcap A'_n$ and by replacing every occurrence of the introduced symbols $A'_1, \ldots, A'_n$ with A. Then $\mathcal{M}$ is a depleting Σ-module of $\mathcal{T}$.

4 Experiments and Evaluation

We implemented the algorithm presented in Figure 1 and the refinement for acyclic TBoxes with repeated concept inclusions in the AMEX system which is written in Java aided by the OWL-API library [5] for ontology manipulation. The inseparability check was implemented using the reduction to the validity of Quantified Boolean Formulae (QBF) and uses the QBF solver sKizzo [2].

To evaluate the efficiency of AMEX and the size of the modules computed by AMEX we compare it to $\top\bot^*$ locality-based module extraction [3, 13] as implemented in the OWL-API library version 3.2.4.1806 (called STAR-modules for ease of pronunciation).

To evaluate the performance of both approaches we consider random and axiom signatures. To generate a random signature size n given a TBox $\mathcal{T}$ we take the set of all concepts in $\mathcal{T}$, i.e. $\mathsf{sig}(\mathcal{T}) \cap \mathsf{N_C}$ and select at random n symbols from this set. For each concept signature size we also include a percentage of role names randomly selected from $\mathsf{sig}(\mathcal{T})$, varying between 0% which equates to just using a concept signature to 100% which would be equal to $\Sigma \cup (\mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R})$. For experiments on axiom signatures, for a given number m, we select at random m axioms from $\mathcal{T}$ and then extract a module for each of the signatures of selected axioms.

In our experiments we used the NCI Thesaurus version 08.09d taken from the Bioportal [15] repository. This version of NCI contains 116 515 logical axioms among which 87 934 are concept inclusions of the form $A \sqsubseteq C$ and 10 366 are concept equations of the form $A \equiv C$. In what follows, $\mathsf{NCI}^\star(\sqsubseteq)$ denotes the TBox consisting of all such inclusions, $\mathsf{NCI}^\star(\equiv)$ denotes the TBox consisting of all such equations, and $\mathsf{NCI}^\star$ denotes the union of both. All three TBoxes are acyclic (with repeated concept inclusions), so AMEX can be applied to them. $\mathsf{NCI}^\star$ together with the rest of the ontology

Role%	0%			25%			50%			75%			100%		
$\|\Sigma\|$	Star	AMEX	% Diff	Star	AMEX	% Diff	Star	AMEX	% Diff	Star	AMEX	% Diff	Star	AMEX	% Diff
NCI*															
100	3835.7	676.6	467%	3848.6	943.7	308%	3891.7	984.0	295%	3929.4	1014.7	287%	3929.8	1016.5	287%
250	5310.2	1725.9	208%	5365.6	1795.2	199%	5463.1	1871.5	192%	5506.3	1919.3	187%	5505.4	1918.0	187%
500	6985.9	2735.9	155%	7109.6	2844.9	150%	7165.5	2930.3	145%	7252.8	3002.1	142%	7245.9	2990.1	142%
750	8223.3	3572.7	130%	8355.2	3698.8	126%	8464.4	3806.1	122%	8538.5	3878.7	120%	8526.1	3872.0	120%
1000	9276.7	4333.6	114%	9397.2	4458.4	111%	9492.8	4573.9	108%	9564.9	4627.1	107%	9565.3	4642.7	106%
NCI* ($\sqsubseteq$)															
100	55.47	65.04	-15%	232.76	281.90	-17%	286.13	318.81	-10%	312.59	333.65	-6%	339.83	351.70	-3%
250	328.28	390.81	-16%	559.56	657.37	-15%	651.05	718.62	-9%	712.87	759.06	-6%	765.30	796.47	-4%
500	852.89	1007.34	-15%	1046.44	1190.43	-12%	1193.75	1301.77	-8%	1278.48	1355.05	-6%	1378.30	1435.99	-4%
750	1325.96	1541.33	-14%	1517.68	1692.20	-10%	1675.37	1808.43	-7%	1802.61	1905.29	-5%	1921.13	1993.60	-4%
1000	1786.342	2039.67	-12%	1973.82	2174.04	-9%	2157.33	2314.21	-7%	2299.00	2416.32	-5%	2440.76	2527.34	-3%
NCI* ($\equiv$)															
100	2784.33	316.31	780%	2792.99	319.73	774%	2785.51	318.49	775%	2770.32	318.61	770%	2779.03	318.43	773%
250	3982.18	622.74	539%	3988.51	626.22	537%	3984.76	624.03	539%	3989.88	624.62	539%	3982.62	625.91	536%
500	4975.97	1001.20	397%	4988.08	1003.83	397%	4984.67	1002.06	397%	4983.22	1004.13	396%	4988.71	1004.00	397%
750	5529.94	1309.98	322%	5540.59	1315.34	321%	5533.68	1309.22	323%	5532.04	1310.77	322%	5531.00	1311.72	322%
1000	5899.871	1577.42	274%	5897.36	1576.94	274%	5891.82	1576.71	274%	5894.13	1574.57	274%	5900.37	1578.06	274%

Fig. 3. Random signature comparison

(18 215 axioms) is called NCI and contains, in addition, role inclusions, domain and range restrictions, disjointness axioms, data properties, and 17 763 ABox assertions.

The majority of NCI* (all but 4 588 axioms) are $\mathcal{EL}$-inclusions. The non-$\mathcal{EL}$ inclusions contain 7 806 occurrences of value restrictions. The signature of NCI* contains 68 862 concept and 88 role names.

Experiments with NCI* and its Fragments The results given in Figure 3 show the average sizes (over 1 000 random signatures for each signature size and role percentage combination) of the modules computed by the two approaches for random signatures. It can be seen that

- in NCI$^\star(\equiv)$, AMEX-modules are significantly smaller than STAR-modules (between 270% and 780%);
- in NCI$^\star(\sqsubseteq)$, STAR-modules are, on average, slightly smaller than AMEX modules;
- in NCI*, AMEX-modules are still significantly smaller than STAR-modules, but less so than in NCI$^\star(\equiv)$.

The huge difference between modules in NCI$^\star(\sqsubseteq)$ and NCI$^\star(\equiv)$ can be explained as follows: it is shown in [8] that for acyclic $\mathcal{EL}$-TBoxes without concept equations, AMEX-modules and STAR-modules coincide. This is not the case for acyclic $\mathcal{ALCQI}$-TBoxes (there can be axioms in STAR-modules that are not AMEX-modules and vice versa), but since the vast majority of axioms in NCI$^\star(\sqsubseteq)$ are $\mathcal{EL}$-inclusions one should not expect any significant difference between the two types of modules. Thus, it is exactly those acyclic TBoxes that contain many concept equations for which AMEX-modules are significantly smaller than STAR-modules (see Example 1 for an illustration).

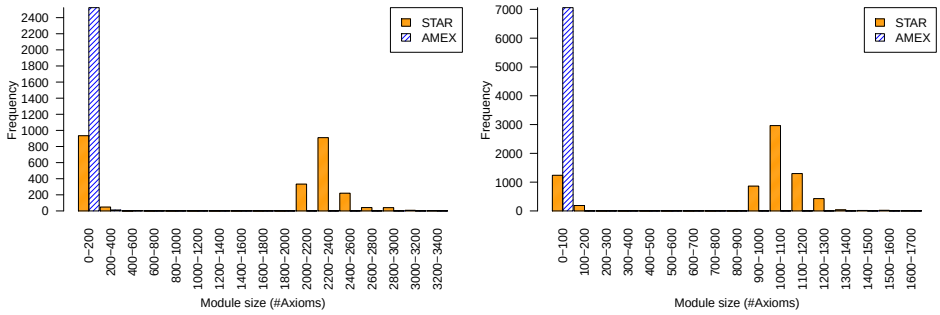


Fig. 4. Frequency of module sizes for NCI* (left) and NCI$^\star(\equiv)$ (right).

Figure 4 summarises our experimental results for modules extracted for axiom signatures. The figure shows the frequency of AMEX and STAR-modules of a given size within NCI* and NCI$^\star(\equiv)$ for the cases when the modules differ – which is in 13% and 68% of cases, respectively. For NCI$^\star(\equiv)$ in the cases in which we find a difference the STAR module is always larger than the corresponding AMEX module with an average difference of 865.6 axioms. For NCI* in a few (87 cases) the STAR modules are smaller than the corresponding AMEX ones by an average difference of 6.9 axioms whereas in the rest of the cases the STAR modules are much larger with an average difference of 1427 axioms. We do not show the results for NCI$^\star(\sqsubseteq)$ since, as explained above for the experiments with random signatures, there is essentially no difference between AMEX and STAR-modules.

These experiments were carried out on a PC with an Intel i5 CPU @ 3.30GHz with 2GB of Java heap space available to the program. For NCI* the average time taken per extraction was just under 3s and the maximum time taken was 15s. Interestingly, in almost all experiments the QBF solver was called just once. Thus, in most cases the modules were computed purely syntactically and the QBF solver simply provided an assurance that the extracted axioms indeed constituted a depleting module. Only in 3% of all experiments the QBF solver identified separability causing axioms. The maximal number of separability causing axioms recorded in any single extraction was 4 and the maximal number of QBF solver calls themselves was 73.

Experiments with full NCI Although AMEX-modules are significantly smaller than STAR-modules for acyclic TBoxes containing many concept equations, the applications of AMEX alone are very limited since most ontologies contain additional axioms such as disjointness axioms, role inclusions, and domain and range restrictions. To tackle this problem we first observe that, in principle, AMEX can be applied to any general TBoxes: given such a TBox $\mathcal{T}$, one can split $\mathcal{T}$ into two parts $\mathcal{T}_1$ and $\mathcal{T}_2$, where $\mathcal{T}_1$ is an acyclic $\mathcal{ALCQI}$-TBox (and as large as possible) and $\mathcal{T}_2 := \mathcal{T} \setminus \mathcal{T}_1$. Then for any signature Σ it follows from the robustness properties [7] of the inseparability relation $\equiv_\Sigma$ that if $\mathcal{M}$ is a depleting $\Sigma \cup \mathsf{sig}(\mathcal{T}_2)$-module of $\mathcal{T}_1$ (note that $\mathcal{M}$ can be

computed by AMEX), then $\mathcal{M} \cup \mathcal{T}_2$ is a depleting Σ-module of $\mathcal{T}$ as well. Such a direct application of AMEX to general TBoxes is unlikely to compute small modules when $\mathcal{T}_2$ is large. However, our first experimental results suggest that this approach is beneficial when iterated with STAR-module extraction. The following result provides the theoretical underpinning for our experiments.

Theorem 3. *Let $\mathcal{M} \subseteq \mathcal{M}' \subseteq \mathcal{T}$ be TBoxes and Σ a signature such that $\mathcal{M}'$ is a depleting Σ-module of $\mathcal{T}$ and $\mathcal{M}$ is a depleting Σ-module of $\mathcal{M}'$. Then $\mathcal{M}$ is a depleting Σ-module of $\mathcal{T}$.*

Since both AMEX and STAR compute depleting Σ-modules, given a signature Σ and ontology $\mathcal{T}$ one can extract an AMEX module from the STAR module (and vice versa) and have the guarantee the resulting module is still a depleting Σ-module of $\mathcal{T}$. In this way, one can repeatedly extract from the output of one extraction approach again a module using the other approach until the sequence of modules becomes stable.

The following experiments are based on a naïve implementation of this hybrid approach and extract modules from the full version of NCI. Again we consider random concept signatures with varying amount of role names. The experiments shown in Figure 5 are based on 200 signatures for each concept signature size/role percentage combination and compare the average size of modules extracted using the hybrid approach and using STAR extraction only.

Role%	0%			25%			50%			75%			100%		
$\lvert\Sigma\rvert$	Star	Iterated	% Diff	Star	Iterated	% Diff	Star	Iterated	% Diff	Star	Iterated	% Diff	Star	Iterated	% Diff
100	5385.7	1949.5	176%	9569.8	6177.7	55%	13733.8	10339.0	33%	19486.4	16089.1	21%	23196.6	19810.2	17%
250	7298.6	3268.7	123%	11959.8	7963.9	50%	16072.1	12069.6	33%	20974.9	16978.8	24%	25141.0	21134.7	19%
500	9445.0	4827.6	96%	13165.1	8533.4	54%	16406.7	11767.0	39%	23046.8	18418.3	25%	27331.2	22691.9	20%
750	11070.2	6058.6	74%	15268.3	10235.9	49%	19696.2	14683.3	34%	23705.7	18689.6	27%	28917.3	23903.4	21%
1000	12370.7	7108.5	74%	16434.7	11174.0	47%	21978.6	16737.6	31%	25529.0	20286.5	26%	30218.5	24965.4	21%

Fig. 5. Iterative module extraction from NCI

For all signatures we found a reduction in the size of the module when iterated with the STAR module on its own being between 17% and 176% larger than the hybrid module.

In Figure 6, we show the results of our experiments for axioms signatures. They are based on 20 000 randomly selected axioms from the full NCI Thesaurus. 13% of such signatures showed a difference from the STAR module. The frequency of module sizes for the cases when the modules differ is given in Figure 6. The average difference in size, for the cases when there is a difference, is 295.2 axioms.

All individual extractions using the hybrid approach saw exactly 2 alternations of the STAR module extraction whereas the AMEX extraction varied between 1 and 2 times. The cases in which the AMEX extraction alternated just once happened much more often as the signature sizes grew and the difference between the respective module sizes became smaller. The additional time taken to extract the hybrid module compared to the STAR extraction alone was at most only 2.2 seconds.

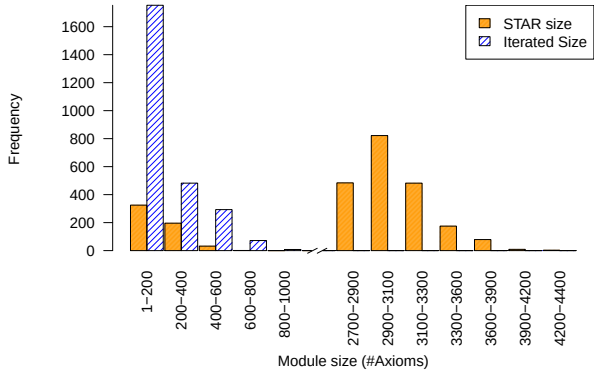


Fig. 6. Frequency of module sizes

5 Conclusion

We have presented a new system, AMEX, for depleting module extraction from acyclic $\mathcal{ALCQI}$-TBoxes. Using the NCI Thesaurus, we have compared the size of AMEX-modules with the size of $\top\bot^*$-modules computed by the OWL-API library implementation (referred as STAR-modules) and we have presented a hybrid approach in which STAR and AMEX-module extraction are used iteratively. The results show that for TBoxes with many axioms of the form $A \equiv C$, AMEX-modules can be significantly smaller than STAR-modules and that an iterative approach can lead to significantly smaller modules than 'pure' STAR-modules. In contrast to [4], where a large number of ontologies are used to compare STAR-modules and MEX-modules we consider NCI only. The reason is that the majority of ontologies considered in [4] contain no (or only a very small set) of axioms of the form $A \equiv C$ that form an acyclic subset of the ontology. For such ontologies it follows both from theoretical results in [8] and experimental results in [4] that there is no significant difference between AMEX and STAR-modules. Instead, we focus on a high quality ontology with a reasonable number of concept equations and where theory predicts that minimal depleting modules can be much smaller than STAR-modules. Many research questions remain to be explored. In particular, to apply AMEX to a larger class of ontologies in an iterative approach, one has to generalise the notion of acyclic TBoxes in such a way that the underpinning methodology of AMEX can still be generalised.

References

1. F. Baader, D. Calvanes, D. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, Cambridge, UK, 2003.

2. M. Benedetti. sKizzo: a QBF decision procedure based on propositional skolemization and symbolic reasoning. Technical Report 04-11-03, ITC-irst, 2004.
3. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: theory and practice. *Journal of Artificial Intelligence Research (JAIR)*, 31:273–318, 2008.
4. C. Del Vescovo, P. Klinov, B. Parsia, U. Sattler, T. Schneider, and D. Tsarkov. Empirical study of logic-based modules: Cheap is cheerful. Technical report, University of Manchester, 2013.
5. M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011.
6. B. Konev, R. Kontchakov, M. Ludwig, T. Schneider, F. Wolter, and M. Zakharyaschev. Conjunctive query inseparability of OWL 2 QL TBoxes. In *Proceedings of the 25th Conference on Artificial Intelligence, AAAI 2011*, pages 221–226, Menlo Park, CA, USA, 2011. AAAI Press.
7. B. Konev, C. Lutz, D. Walther, and F. Wolter. Formal properties of modularisation. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*, pages 25–66. Springer, Berlin, Heidelberg, 2009.
8. B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203:66–103, 2013.
9. R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev. Minimal module extraction from DL-Lite ontologies using QBF solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, pages 836–841, Menlo Park, CA, USA, 2009. AAAI Press.
10. R. Kontchakov, F. Wolter, and M. Zakharyaschev. Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Artificial Intelligence*, 174(15):1093–1141, 2010.
11. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computing*, 45(2):194–228, 2010.
12. R. Nortjé, K. Britz, and T. Meyer. Module-theoretic properties of reachability modules for sriq. In *Proceedings of the 26th international workshop on description logic, DL 2013*, CEUR Workshop Proceedings. CEUR-WS.org, 2013.
13. U. Sattler, T. Schneider, and M. Zakharyaschev. Which kind of module should I extract? In *Proceedings of the 22nd International Workshop on Description Logics, DL 2009*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
14. H. Stuckenschmidt, C. Parent, and S. Spaccapietra, editors. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2009.
15. P. L. Whetzel, N. F. Noy, N. H. Shah, P. R. Alexander, C. Nyulas, T. Tudorache, and M. A. Musen. Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Research*, 39(Web-Server-Issue):541–545, 2011.

Towards fast Atomic Decomposition using Axiom Dependency Hypergraphs*

Francisco Martín-Recuerda[1] and Dirk Walther[2]

[1] Universidad Politécnica de Madrid, Spain
fmartinrecuerda@fi.upm.es
[2] TU Dresden, Germany
Center for Advancing Electronics Dresden
dirk.walther@tu-dresden.de

Abstract. Atomic decomposition of ontologies has been suggested as a tool to understand the modular structure of ontologies. It consists of a polynomial size representation of potentially exponentially many modules of an ontology. Tractable algorithms for computing the atomic decomposition for locality-based modules have been introduced, albeit leaving room for improvement in terms of running time. In this paper, we consider ontologies formulated in OWL-EL. We introduce a notion of an axiom dependency hypergraph for an ontology, which represents how axioms are included in locality-based modules. We use a standard algorithm from graph theory to compute strongly connected components of the axiom dependency hypergraph and show that such components correspond to atoms of the ontology. An empirical evaluation of the algorithm on large fragments of biomedical ontologies confirms a significant improvement in running time.

1 Introduction

The atomic decomposition of an ontology provides a compact representation of all possible modules that can be extracted from a given ontology. It represents a significant contribution to a better understanding of the internal structure of the ontology.

In this paper, we introduce a novel representation model of OWL-EL ontologies based on directed hypergraphs that explicitly represent information for syntactic locality and atomic decomposition. We call this model *Axiom Dependency Hypergraphs* (ADHs). A directed hypergraph is a generalisation of a directed graph in which edges (in this case hyperedges) can connect two nonempty disjoint sets of nodes (or vertices). The nodes in such hypergraphs represent the axioms of an ontology, and hyperedges indicate subset relations between terms

* We thank the reviewers of the workshop WoMO 2013 for their comments. The first author acknowledges the support of the EU project SEALS (FP7-ICT-238975), and the second author the support of the German Research Foundation (DFG) within the Cluster of Excellence 'Center for Advancing Electronics Dresden'.

in axioms. By using standard hyperpath search algorithms it is possible to efficiently extract locality-based modules (in linear time) and to compute the atomic decomposition of the ontology.

The use of hypergraph-based models for locality-based module extraction is not new [8] and it has been recently further extended [6]. However, this is the first time that it has been applied for improving existent algorithms for atomic decomposition of OWL ontologies. We compare a prototypical implementation of our hypergraph model against the fastest implementation of atomic decomposition that we are aware of [10]. Our experiments confirm a significant improvement in running time for (certain fragments of) biomedical ontologies.

The paper is organised as follows. In Section 2, we review the main notions on syntactic $\bot$-locality, atomic decomposition, and hypergraphs. In Section 3, we introduce axiom dependency hypergraphs and discuss their size, and evaluate the performance of atomic decomposition based on these graphs in Section 4. We conclude this paper in a final section.

2 Preliminaries

In this paper, we consider ontologies formulated in the description logic $\mathcal{EL}$, which allows for conjunction and existential restrictions. Large parts of, e.g., biomedical ontologies are expressed in $\mathcal{EL}$. For notions on description logics, we refer to [3] and for $\mathcal{EL}$ to [2]. Moreover, we consider locality-based modules. The notion of locality refers to axioms, i.e., axioms are either local or non-local, and it depends on a signature. A module of an ontology consists of the non-local axioms wrt. a signature. There are two flavours the locality notion comes in: semantic and syntactic locality. Intuitively, an axiom is semantically local wrt. a signature Σ if it does not say anything about the terms in Σ.[1] Checking whether an axiom is semantically local can be computationally expensive as it requires reasoning.[2] The notion of syntactic locality was introduced to allow for efficient computation. Checking for syntactic locality involves checking that an axiom is of a certain form, no reasoning is needed. On the downside, syntactic locality is merely an approximation of semantic locality: all syntactically local axioms are also semantically local, but not *vice versa*. Modules based on syntactic locality can be larger as they contain the modules based on semantic locality and possibly more axioms. We refer to [5] for a more extensive introduction to locality-based modularity.

The notion of $\bot$-locality depends on whether or not a given signature covers all symbols occurring on the LHS of a general concept inclusion, or occurring on the LHS or RHS of a general concept equation. The following proposition makes that precise.[3]

[1] For sets of axioms with this property the term 'safety' is used in the literature [5].

[2] Reasoning with OWL2 is 2-NExpTime-complete in the worst case, but also reasoning with a tractable logic such as $\mathcal{EL}$ can be seen as too difficult in some cases, depending on the size of the input and the application requirements.

[3] The notion of reachability-based modules uses a similar property; see Def. 37 in [8].

Proposition 1. *Let α be an $\mathcal{EL}$-axiom. Let Σ be a signature. Then:*

(i) if $\alpha = (C \sqsubseteq D)$, then α is not syntactic $\bot$-local wrt. Σ iff $\mathsf{sig}(\mathsf{LHS}(\alpha)) \subseteq \Sigma$;
(ii) if $\alpha = (C \equiv D)$, then α is not syntactic $\bot$-local wrt. Σ iff $\mathsf{sig}(\mathsf{LHS}(\alpha)) \subseteq \Sigma$ or $\mathsf{sig}(\mathsf{RHS}(\alpha)) \subseteq \Sigma$. ⊣

The following example illustrates the notion of $\bot$-locality of axioms.

Example 1. Let α and β be the axioms:[4]

$$\alpha := A_1 \sqcap \exists r_1.(A_2 \sqcap \exists r_2.A_3) \sqsubseteq \exists r_3.A_4$$
$$\beta := A_1 \equiv A_2 \sqcap \exists r_1.A_3$$

Axiom α is syntactic $\bot$-local wrt. Σ if any of the symbols occurring in $\mathsf{LHS}(\alpha)$ is not contained in Σ, i.e., if α satisfies any of the following conditions:

- $A_i \notin \Sigma$, for some $i \in \{1, 2, 3\}$; or
- $r_1 \notin \Sigma$ or $r_2 \notin \Sigma$.

Axiom β is syntactic $\bot$-local wrt. Σ if there are two symbols, one occurring in $\mathsf{LHS}(\beta)$ and one in $\mathsf{RHS}(\beta)$, that are not contained in Σ, i.e., if β satisfies any of the following conditions:

- $A_1 \notin \Sigma$ and $A_i \notin \Sigma$, for some $i \in \{2, 3\}$; or
- $A_1 \notin \Sigma$ and $r_1 \notin \Sigma$. ◁

2.1 Extracting modules based on locality

This is the module extraction algorithm from [5], where x stands for any of the semantic and syntactic locality notions, i.e. $x \in \{\emptyset, \Delta, \bot, \top\}$. The algorithm runs in time quadratic in the size of the ontology and signature.

---

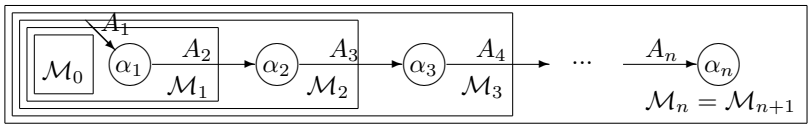
function $\mathsf{Mod}^x_{\mathcal{O}}(\Sigma)$ returns x-local module of $\mathcal{O}$ wrt. Σ
1: $m := 0$
2: $\mathcal{M}_m := \emptyset$
3: do
4: $\quad m := m + 1$
5: $\quad \mathcal{M}_m := \{\alpha \in \mathcal{O} \mid \alpha$ is not x-local wrt. $\Sigma \cup \mathsf{sig}(\mathcal{M}_{m-1})\}$
6: until $\mathcal{M}_m = \mathcal{M}_{m-1}$
7: return $\mathcal{M}_m$

The algorithm takes an ontology $\mathcal{O}$ and a signature Σ as input and returns a subset $\mathcal{M}$ of $\mathcal{O}$. The computed set $\mathcal{M}$ consists of axioms that are not x-local wrt. $\Sigma \cup \mathsf{sig}(\mathcal{M})$, or equivalently, $\mathcal{O} \setminus \mathcal{M}$ consists of axioms that are local wrt. $\Sigma \cup \mathsf{sig}(\mathcal{M})$. The algorithm produces a finite sequence $\mathcal{M}_0, \ldots, \mathcal{M}_n$, $n \geq 0$, of subsets of the ontology $\mathcal{O}$, where $\mathcal{M}_n$ is the $\bot$-local module of $\mathcal{O}$ wrt.

[4] These axioms are of the forms that occur frequently in the biomedical ontology Full-Galen.

$\Sigma \cup \mathsf{sig}(\mathcal{M}_n)$ with Σ being the initial signature. This sequence induces a relation on $\mathcal{O}$ representing the successive inclusion of the axioms into the module. Later we will represent this relation as hyperedges in an axiom dependency hypergraph. This is illustrated in the following example.

Example 2. The signature increases round-by-round due to axioms that are added to the module (line 5). In fact, the RHSs of the axioms introduce new symbols to the signature. Let $\alpha_i = A_i \sqsubseteq A_{i+1}$, for $i \in \{1, ..., n\}$. Let $\mathcal{O} = \{\alpha_1, ..., \alpha_n\}$ and $\Sigma = \{A_1\}$. We run the algorithm on the input $\mathcal{O}$ and Σ. In the first round of the do-until loop (lines 3-6), the axiom α_1 is the only axiom in $\mathcal{O}$ that is not $\bot$-local wrt. $\Sigma \cup \mathsf{sig}(\mathcal{M}_0)$ (line 5), where $\mathcal{M}_0 = \emptyset$ (line 2). So α_1 is added to $\mathcal{M}_1$. In the second round, $\bot$-locality is checked for the extended signature $\Sigma \cup \mathsf{sig}(\mathcal{M}_1) = \{A_1, A_2\}$. Axioms α_1 and α_2 are now non-$\bot$ local wrt. $\Sigma \cup \mathsf{sig}(\mathcal{M}_1)$ and both are added to $\mathcal{M}_2$. The algorithm proceeds this way until all axioms of $\mathcal{O}$ are added to $\mathcal{M}_n$. As no more axioms are added the algorithm exits the do-until loop (line 6) and returns $\mathcal{M}_{n+1} = \mathcal{O}$ as the $\bot$-local module of $\mathcal{O}$ wrt. Σ (line 7). The following picture shows the module extraction process as a graph $G_{\mathsf{mod}^\bot(\mathcal{O},\Sigma)} = (V, E)$, where $V = \{\alpha_1, ..., \alpha_n\}$ and $E = \{(\alpha_i, A_{i+1}, \alpha_{i+1}) \mid i \in \{1, ..., n-1\}\}$, and the sequence $\mathcal{M}_0, ..., \mathcal{M}_n, \mathcal{M}_{n+1}$ produced by the module extraction algorithm. Additionally we indicate with a tilted arrow labelled with A_1 to node α_1 that the concept name A_1 is provided by the initial signature. In this case, α_1 is the first axiom to be included by the module extraction algorithm.



◁

2.2 Atomic decomposition

Atoms represent sets of highly related axioms in the sense that they always co-occur in modules. A set $\mathfrak{a}$ of axioms from an ontology $\mathcal{O}$ is an *atom of* $\mathcal{O}$ if for every module $\mathcal{M}$ of $\mathcal{O}$, it holds that $\mathfrak{a} \subseteq \mathcal{M}$ or $\mathfrak{a} \cap \mathcal{M} = \emptyset$. We denote with $\mathsf{Atoms}_\mathcal{O}$ the set of all atoms of $\mathcal{O}$. The atoms of an ontology partition the set of its axioms (i.e., each axiom occurs in exactly one atom). A dependency relation between pairs of atoms can be established: an atom $\mathfrak{a}_2$ *depends on* an atom $\mathfrak{a}_1$ in an ontology $\mathcal{O}$ (written $\mathfrak{a}_1 \succcurlyeq_\mathcal{O} \mathfrak{a}_2$) if $\mathfrak{a}_2$ occurs in every module of $\mathcal{O}$ containing $\mathfrak{a}_1$. For a given ontology, the pair $\langle \mathsf{Atoms}_\mathcal{O}, \succcurlyeq_\mathcal{O} \rangle$ consisting of the set of atoms together with the dependency relation between them is called *Atomic Decomposition*.[5] It provides a compact representation of all modules of an ontology as every module is composed of the axioms of certain atoms. Therefore, the atomic decomposition contributes to a better understanding of the internal structure of ontologies with possible implications in ontology engineering and reasoner design. The representation of the modules in terms of atoms and their

[5] Here we use atomic decomposition for $\bot$-local modules denoted as $\langle \mathsf{Atoms}^\bot_\mathcal{O}, \succcurlyeq^\bot_\mathcal{O} \rangle$.

interdependencies is of linear size and it can be computed in polynomial time, whereas there are exponentially many possible modules of an ontology. However, not *all* modules of an ontology can be computed from its atomic decomposition. The atomic decomposition was first introduced in [11].

2.3 Directed hypergraphs

A *directed hypergraph* [4] is a tuple $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a non-empty set of *nodes* (vertices), and $\mathcal{E}$ is a set of *hyperedges* (*hyperarcs*). A *hyperedge* e is also defined as a pair $(T(e), H(e))$, where $T(e)$ and $H(e)$ are nonempty disjoint subsets of $\mathcal{V}$. $H(e)$ ($T(e)$) is known as the *head* (*tail*) and represents a set of nodes where the hyperedge ends (starts). For each node $v \in H(e)$ ($v \in T(e)$), e is an *incoming* (*outgoing*) hyperedge to v. A *hyperpath* from node v_1 to node v_n is a sequence of the form $P(v_1, v_n) = v_1 e_1 v_2 e_2 \cdots e_{n-1} v_n$ such that $v_i \in T(e_i)$ and $v_{i+1} \in H(e_i)$ for every $i \in \{1, ..., n-1\}$. In addition, if node $v_n \in T(e_1)$, the hyperpath $P(v_1, v_n)$ is a *hypercycle*.

A node v_2 is *B-connected*[6] (or forward reachable[7]) from v_1 if (i) $v_2 = v_1$ (v_2 is B-connected from itself), or (ii) there is a hyperedge e such that $v_2 \in H(e)$ and all the elements of $T(e)$ are B-connected from v_1. A *B-hyperpath* from node v_1 to node v_2 in a hypergraph $\mathcal{H}$ is a minimal (in terms of hyperedges and nodes) subhypergraph of $\mathcal{H}$ such that v_2 is B-connected to v_1.

In a directed hypergraph $\mathcal{H}$, two nodes v_1 and v_2 are *strongly B-connected* if v_2 is B-connected to v_1 and *vice versa*. In other words, both nodes, v_1 and v_2, are *mutually reachable*. A *strongly B-connected component (SCC)* is a set of nodes from $\mathcal{H}$ which are all are mutually reachable [1].[8] Note that two nodes that are mutually reachable in a hypergraph belong to the same hypercycle.

In the case of directed graph, it is possible to calculate all the strongly connected components in linear time [9, 7]. Unfortunately, none of these linear time algorithms can be easily adapted to directed hypergraphs due to the more complicated notion of reachability in hypergraphs [1]. For instance, while in a directed graph the last node of a path can be reached from any other node in the path, this may not be the case in a hyperpath [1].

3 Axiom dependency hypergraphs

Axiom dependency hypergraphs are a representation model for ontologies based on directed hypergraphs that explicitly represent information for locality and atomic decomposition. The nodes in such graphs represent the axioms of an ontology, and hyperedges indicate subset relationships between terms in axioms

[6] There is a similar notion called *F-connectivity* [4].

[7] We walk on a hypergraph from the tails to the heads of the hyperedges. If all nodes in the tail of a hyperedge have been reached it is possible to reach all the nodes of the head of the hyperedge.

[8] Notice that an SCC can be a singleton set as the reachability relation is reflexive, i.e., any axiom is mutually reachable from itself.

(cf. Proposition 1). By using standard hyperpath search algorithms it is possible to efficiently extract locality based modules (in linear time) and to compute the atomic decomposition of the ontology.

The $\bot$-*locality signatures* $\bot\text{-}\mathsf{sig}(\alpha)$ of an axiom α is the set of signatures

$$\bot\text{-}\mathsf{sig}(\alpha) = \begin{cases} \{\mathsf{sig}(\mathsf{LHS}(\alpha))\} & \text{if } \alpha \text{ is of the form } C \sqsubseteq D; \\ \{\mathsf{sig}(\mathsf{LHS}(\alpha)), \mathsf{sig}(\mathsf{RHS}(\alpha))\} & \text{if } \alpha \text{ is of the form } C \equiv D. \end{cases}$$

Intuitively, $\bot\text{-}\mathsf{sig}(\cdot)$ is a function assigning to an axiom α the signatures $\bot\text{-}\mathsf{sig}(\alpha)$ that are relevant for deciding the $\bot$-locality status of α. More precisely, we have that α is not $\bot$-local wrt. S, for every $S \in \bot\text{-}\mathsf{sig}(\alpha)$.

Axiom dependency hypergraphs for $\mathcal{EL}$-ontologies are defined as follows.

Definition 1 (Axiom dependency hypergraph for $\bot$-locality). *Let $\mathcal{O}$ be an $\mathcal{EL}$-ontology. The* $\bot$-locality axiom dependency hypergraph (ADH) $\mathcal{H}^{\bot}_{\mathcal{O}}$ for $\mathcal{O}$ *is defined as* $\mathcal{H}^{\bot}_{\mathcal{O}} = (\mathcal{V}, \mathcal{E})$*, where*

- $\mathcal{V} = \mathcal{O}$*; and*
- $e = (T(e), H(e)) \in \mathcal{E}$ *iff the following holds:*
  - *(i)* $H(e) = \{\beta\}$*, for some* $\beta \in \mathcal{V}$*, and*
  - *(ii)* $T(e) \subseteq \mathcal{V} \setminus \{\beta\}$ *such that* $|T(e)|$ *is minimal with the property* $S \subseteq \mathsf{sig}(T(e))$*, for some* $S \in \bot\text{-}\mathsf{sig}(\beta)$*.*

$\dashv$

Note that the axiom dependency hypergraph for an $\mathcal{EL}$-ontology is defined for the notion of $\bot$-locality.[9] The nodes are the axioms of the ontology, and the hyperedges are associating axioms $\alpha_1, \ldots, \alpha_n$ with a single axiom β such that one of the $\bot$-locality signatures of β is contained in the signature of the axioms α_i. The minimality condition states that each of the axioms α_i is required and none of them is superfluous for covering some of the $\bot$-locality signatures of β.[10]

Example 3. Let $\mathcal{O} = \{\alpha_1, ..., \alpha_5\}$, where

$$\begin{array}{lll} \alpha_1 := A \sqsubseteq B & \alpha_3 := E \sqsubseteq A \sqcap C \sqcap D & \alpha_5 := X \sqsubseteq A \\ \alpha_2 := B \sqcap C \sqcap D \sqsubseteq E & \alpha_4 := A \sqsubseteq X & \end{array}$$

The ADH $\mathcal{H}^{\bot}_{\mathcal{O}}$ contains the hyperedges $e_1 = (\{\alpha_1, \alpha_3\}, \{\alpha_2\})$, $e_2 = (\{\alpha_1\}, \{\alpha_4\})$, $e_3 = (\{\alpha_2\}, \{\alpha_3\})$, $e_4 = (\{\alpha_3\}, \{\alpha_1\})$, $e_5 = (\{\alpha_3\}, \{\alpha_4\})$, $e_6 = (\{\alpha_4\}, \{\alpha_1\})$, $e_7 = (\{\alpha_4\}, \{\alpha_5\})$, $e_8 = (\{\alpha_5\}, \{\alpha_1\})$ and $e_9 = (\{\alpha_5\}, \{\alpha_4\})$; see Example 4 for a visualisation of the graph. Now obtain $\mathcal{O}'$ by replacing α_2 with $\alpha'_2 := B \sqcap C \sqcap D \equiv E$. Then the ADH $\mathcal{H}^{\bot}_{\mathcal{O}'}$ contains one additional edge $e_{10} = (\{\alpha_3\}, \{\alpha_2\})$. $\lhd$

B-connectivity (forward reachability) in axiom dependency hypergraphs can be used to specify $\bot$-local modules in the corresponding ontology. Denote with $\mathsf{Mod}^{\bot}_{\mathcal{O}}(\alpha)$ the $\bot$-local module of ontology $\mathcal{O}$ wrt. the signature of axiom α.

[9] Axiom dependency hypergraphs for the notion of $\top$-locality can be defined in a similar way.

[10] Note that the minimality condition in Def. 1 is to avoid superfluous hyperedges, but it is not required in terms of correctness.

Proposition 2. *Let $\mathcal{O}$ be an $\mathcal{EL}$-ontology and α an $\mathcal{EL}$-axiom. Then: $\mathsf{Mod}^{\perp}_{\mathcal{O}}(\alpha) = \{\beta \mid \alpha$ is B-connected to β in $\mathcal{H}^{\perp}_{\mathcal{O}}\}$.* ⊣

Reachability-based modules have been defined in other hypergraph respresentations of ontologies in [8]. This proposition can be shown similarly.

The set of atoms for an $\mathcal{EL}$-ontology $\mathcal{O}$ corresponds to the set of strongly connected components in the axiom dependency hypergraph for $\mathcal{O}$. Let $\mathsf{SCCs}(\mathcal{H}^{\perp}_{\mathcal{O}})$ be the set of strongly connected components of the hypergraph $\mathcal{H}^{\perp}_{\mathcal{O}}$.

Proposition 3. *Let $\mathcal{O}$ be an $\mathcal{EL}$-ontology. Then: $\mathsf{Atoms}^{\perp}_{\mathcal{O}} = \mathsf{SCCs}(\mathcal{H}^{\perp}_{\mathcal{O}})$.* ⊣

The proposition can readily be seen as follows. As a corollary of Proposition 2, mutually reachable axioms are contained in the same modules, which is equivalent to the axioms forming an atom of the ontology. The nodes in a strongly connected component are all mutually reachable. Thus, both notions are equivalent.

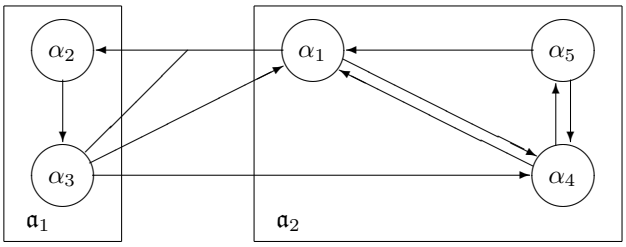
The dependencies between atoms for an $\mathcal{EL}$-ontology $\mathcal{O}$ (as given by the relation $\succcurlyeq^{\perp}_{\mathcal{O}}$) correspond to a certain binary relation over the set of strongly connected components of $\mathcal{H}^{\perp}_{\mathcal{O}}$, which is defined as follows. For $S_1, S_2 \in \mathsf{SCCs}(\mathcal{H}^{\perp}_{\mathcal{O}})$, we say that S_2 *depends* on S_1 (written $S_1 \rightarrow^{\perp}_{\mathcal{O}} S_2$) if there is an hyperedge $e = (T(e), H(e))$ in $\mathcal{H}^{\perp}_{\mathcal{O}}$ such that $T(e) \subseteq S_1$ and $H(e) \subseteq S_2$. Let $\rightarrow^{\perp}_{\mathcal{O}}{}^{*}$ be the reflexive, transitive closure of $\rightarrow^{\perp}_{\mathcal{O}}$.

Proposition 4. *Let $\mathcal{O}$ be an $\mathcal{EL}$-ontology. Let $\mathfrak{a}_1, \mathfrak{a}_2 \in \mathsf{Atoms}^{\perp}_{\mathcal{O}}$ and $S_1, S_2 \in \mathsf{SCCs}(\mathcal{H}^{\perp}_{\mathcal{O}})$ such that $\mathfrak{a}_1 = S_1$ and $\mathfrak{a}_2 = S_2$. Then: $\mathfrak{a}_1 \succcurlyeq^{\perp}_{\mathcal{O}} \mathfrak{a}_2$ iff $S_1 \rightarrow^{\perp}_{\mathcal{O}}{}^{*} S_2$.* ⊣

Example 4. Consider again the ontology $\mathcal{O}$ and its axiom dependency hypergraph $\mathcal{H}^{\perp}_{\mathcal{O}}$ from Example 3. We obtain the following $\perp$-local modules for the axioms:

$$\mathsf{Mod}^{\perp}_{\mathcal{O}}(\alpha_1) = \{\alpha_1, \alpha_4, \alpha_5\} \qquad \mathsf{Mod}^{\perp}_{\mathcal{O}}(\alpha_4) = \{\alpha_1, \alpha_4, \alpha_5\}$$
$$\mathsf{Mod}^{\perp}_{\mathcal{O}}(\alpha_2) = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\} \qquad \mathsf{Mod}^{\perp}_{\mathcal{O}}(\alpha_5) = \{\alpha_1, \alpha_4, \alpha_5\}$$
$$\mathsf{Mod}^{\perp}_{\mathcal{O}}(\alpha_3) = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$$

The resulting atoms in $\mathsf{Atoms}_{\mathcal{O}}$ are $\mathfrak{a}_1 = \{\alpha_2, \alpha_3\}$ and $\mathfrak{a}_2 = \{\alpha_1, \alpha_4, \alpha_5\}$, where $\mathfrak{a}_1 \succcurlyeq \mathfrak{a}_2$, i.e., $\mathfrak{a}_2$ depends on $\mathfrak{a}_1$. The ADH $\mathcal{H}^{\perp}_{\mathcal{O}}$ and the atoms of $\mathcal{O}$ can be depicted as follows:



Consider the strongly connected components of $\mathcal{H}^{\perp}_{\mathcal{O}}$. Axiom α_1 is B-connected with the axioms α_4 and α_5, α_4 is B-connected with α_1 and α_5, and α_5 is B-connected with α_1 and α_4. Axiom α_2 is B-connected with α_3 and *vice versa*.

Axioms α_2, α_3 are each B-connected with α_1, α_4 and α_5, but not *vice versa*. Hence, $\{\alpha_1, \alpha_2, \alpha_4\}$ and $\{\alpha_2, \alpha_3\}$ are the strongly connected components of $\mathcal{H}^{\perp}_{\mathcal{O}}$. Moreover, we say that the former component *depends* on the latter as any two axioms contained in them are unilaterally and not mutually B-connected. Note that the atoms $\mathfrak{a}_1$ and $\mathfrak{a}_2$ of $\mathcal{O}$ and their dependency coincide with the strongly connected components of $\mathcal{H}^{\perp}_{\mathcal{O}}$. $\lhd$

For ontologies consisting of axioms of the form $A \sqsubseteq C$, where A is a concept name, the locality dependencies between axioms can be represented in directed graphs (i.e., hypergraphs in which the tail of any hyperedge contains only one axiom). For such ontologies $\mathcal{O}$, the definition of the axiom dependency hypergraph $\mathcal{H}^{\perp}_{\mathcal{O}} = (\mathcal{V}, \mathcal{E})$ can be simplified (cf. Def. 1). The condition for a hyperedge $e = (\{\alpha\}, \{\beta\})$, for $\alpha, \beta \in \mathcal{V}$, to be contained in $\mathcal{E}$ is now:

$$e = (\{\alpha\}, \{\beta\}) \in \mathcal{E} \;\text{ iff }\; \mathsf{sig}(\mathsf{LHS}(\beta)) \subseteq \mathsf{sig}(\alpha)$$

The advantage of this representation is that we can directly use a standard linear time algorithm for calculating strongly connected components of directed graphs [9, 7]. We notice the majority of axioms in biomedical ontologies (that we have considered in the evaluation part of this paper) are axioms of the form $A \sqsubseteq C$. For general ontologies, we can apply a three-phase approach similar to the one suggested in [1] for hypergraphs: first, we compute the strongly connected components for axioms of the form $A \sqsubseteq C$ using a linear-time algorithm; second, we collapse the strongly connected components into single nodes; and, finally, we compute the strongly connected components of the revised axiom dependency hypergraph using the notion of mutual reachability (which can be computed in at most quadratic time [1]).

3.1 Size of axiom dependency hypergraphs (worst-case)

We observe that axiom dependency hypergraphs may be of exponential size.

Proposition 5. *There exists ontologies $\mathcal{O}$ such that the axiom dependency hypergraph $\mathcal{H}_{\mathcal{O}}$ contains exponentially many hyperedges in the number of axioms of $\mathcal{O}$.* $\dashv$

We show the proposition with the following example.

Example 5. Let n, k be two natural numbers with $n > 0$ and $k > 0$. Ontology $\mathcal{O}_{n,k}$ is defined as:

$$\mathcal{O}_{n,k} = \{\alpha := X_1 \sqcap \ldots \sqcap X_n \sqsubseteq Y\} \cup \{\alpha^i_j := A^i_j \sqsubseteq X_i \mid i \in \{1, ..., n\}, j \in \{1, ..., k\}\}$$

Axiom α is the only axiom in $\mathcal{O}_{n,k}$ with a complex LHS, which in turn is a conjunction of n many concept names $X_1, ..., X_n$. The axioms α^i_j state that each concept name X_i subsumes k many concept names $A^i_1, ..., A^i_k$. The corresponding axiom dependency graph $\mathcal{H}_{\mathcal{O}_{n,k}}$ is the the tuple $\mathcal{H}_{\mathcal{O}_{n,k}} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{O}_{n,k}$ and

$$\mathcal{E} = \{(\{\beta_1, ..., \beta_n\}, \{\alpha\}) \mid \beta_i \in \{\alpha^i_1, ..., \alpha^i_k\}, i \in \{1, ..., n\}\}.$$

It can readily be seen that $\mathcal{H}_{\mathcal{O}_{n,k}}$ contains k^n many hyperedges, where n, k are each bound by the number of axioms in $\mathcal{O}_{n,k}$. $\lhd$

The axiom dependency hypergraph serves as a tool to determine the atoms of an ontology via calculating the strongly connected components of the graph. Example 5 illustrates a problem with the size of the axiom dependency hypergraphs as defined in Def. 1. We note, however, that no hyperedge in $\mathcal{H}_{\mathcal{O}_{n,k}}$ is needed for the purpose of determining the atoms for $\mathcal{O}_{n,k}$. For every axiom β in $\mathcal{O}_{n,k}$, the $\bot$-local module $\mathsf{Mod}^{\bot}_{\mathcal{O}}(\beta)$ is a singleton set consisting of β itself. Consequently, the atoms of $\mathcal{O}_{n,k}$ are singleton sets as well as are the strongly connected components of $\mathcal{H}_{\mathcal{O}_{n,k}}$. Note that we obtain the same strongly connected components after removing every hyperedge from $\mathcal{H}_{\mathcal{O}_{n,k}}$. This observation suggests a possible solution to avoid the exponential blow-up problem. The idea is to build the hypergraph using only the "necessary" hyperedges that are required for the computation of the strongly connected components which in turn correspond to atoms. We hypothesize that the hyperedges needed are the ones that preserve the mutual reachability relation between the axioms of the ontology. This means that no more incoming hyperedges are needed per axiom than the total number of axioms in the ontology. We leave a formal treatment of this idea for future work.

3.2 Size of axiom dependency hypergraphs for real ontologies

Primitive concept inclusions, i.e. axioms of the form $A \sqsubseteq C$, where A is a concept name and C a possibly complex concept, are of prime importance for biomedical ontologies. For instance, the majority of axioms in the biomedical ontologies from the Bioportal in the following table are primitive concept inclusions.[11]

Ontology	Signature size	#axioms $A \sqsubseteq C$	percentage of all axioms in ontology	#axioms $C \equiv D$	#role axioms
CPO (12/2011)	136 090	306 111	80.61%	73 461	96
FMA-lite	75 168	119 558	99.99%	0	3
Full-Galen (v1.1)	24 088	25 563	67.81%	9 968	2 165
GO v1.1 (1986)	34 220	61 990	99.99%	0	5
NCBI (v1.2)	847 796	847 755	100%	0	0
RH-MeSH (08/2012)	286 382	403 210	100%	0	0
Snomed CT (2010a)	291 207	227 698	78.18%	63 446	12

Most ontologies contain other axioms as well such as concept equations and role axioms.[12] The table shows the percentage of axioms of the form $A \sqsubseteq C$ to all axioms in each ontology. The ontologies NCBI and RH-Mesh are taxonomies (no logical operators); they consist entirely of axioms of the form $A \sqsubseteq B$, where A, B are concept names.

[11] `http://bioportal.bioontology.org`

[12] Other axiom types are not shown here, e.g., disjointness axioms in CPO.

Ontology $A \sqsubseteq C$-fragment	Signat. size	ADH #nodes	ADH #edges	ADH #SCCs
CPO (12/2011)	136 027	306 111	1 474 962	131 482
FMA-lite	75 141	119 558	1 021 863	54 649
Full-Galen (v1.1)	16 412	25 563	179 770	12 502
GO v1.1 (1986)	34 175	61 990	255 215	34 167
NCBI (v1.2)	847 760	847 755	1 695 474	847 755
RH-MeSH (08/2012)	286 380	403 210	1 435 631	286 263
Snomed CT (2010a)	240 021	227 698	556 474	227 698

The difference between the number of SCCs and the number of nodes in the axiom dependency hypergraph can be seen as a measure of cyclicity of the graph.

The following example illustrates the limitations of axiom dependency graphs. Axioms of the form $C \sqsubseteq D$ or $C \equiv D$ (i.e. axioms whose $\bot$-locality signatures contains more than one symbol) can cause many hyperedges to be included in the axiom dependency hypergraph.

Example 6. Consider this concept equation α from the ontology Full-Galen:

$$\begin{aligned}\mathsf{Incontinence} \equiv\ & \mathsf{Transport} \\ & \sqcap \exists\, \mathsf{actsOn.BodySubstance} \\ & \sqcap \exists\, \mathsf{hasIntentionality.}(\mathsf{Intentionality} \sqcap \exists\, \mathsf{hasAbsoluteState.involuntary}) \\ & \sqcap \exists\, \mathsf{hasUniqueAssociatedDisplacement.}(\mathsf{Displacement} \\ & \qquad \sqcap \exists\, \mathsf{isDisplacementTo.GRAILExteriorOfBody})\end{aligned}$$

The axiom dependency hypergraph $\mathcal{H}^{\bot}_{\text{Full-Galen}}$ contains up to 9.2×10^{12} many hyperedges of the form $e = (T(e), \{\alpha\})$, where $T(e)$ is a set of axioms containing the 12 signature symbols from $\bot\text{-}\mathsf{sig}(\alpha)$.[13] ◁

As indicated in Section 3.1, the number of hyperedges may be reduced while preserving the strongly connected components. The idea is that no more incoming hyperedges are needed per axiom as there are axioms in the ontology. In the case of Full-Galen, we have an upper bound of 37 696 many incoming hyperedges per axiom. In particular, for axiom α from Example 6 we estimate up to 21 095 many hyperedges to preserve the mutual reachability relation involving α.

4 Evaluation

We have implemented a Java program that takes an $\mathcal{EL}$-ontology $\mathcal{O}$ (with axioms of the form $A \sqsubseteq C$) as an input and builds the corresponding axiom dependency hypergraph $\mathcal{H}^{\bot}_{\mathcal{O}}$. Then it computes the set $\mathsf{SCCs}(\mathcal{H}^{\bot}_{\mathcal{O}})$ of strongly connected components of $\mathcal{H}^{\bot}_{\mathcal{O}}$ and the dependencies between these components. We compare the running times of atomic decomposition using FaCT++ with the times needed by the hypergraph-based approach. The following table lists the results.[14]

[13] The number of hyperedges is merely an estimated upper bound (as it does not take into account the minimality condition in Def. 1).

[14] All experiments were performed on an Intel Xeon E5-2640 2.50 GHz with Java version 1.7.0_40 (command line parameters -Xss1G -Xms4G -Xmx8G). We used FaCT++, 64bit, Version 1.6.2 (19 February 2013) and the OWL API version 3.4.4.

Ontology fragment with axioms of the form $A \sqsubseteq C$	time FaCT++	time hypergraph-based approach					speedup
		load ont.	build ADH	comp. SCC	comp. deps.	total	
CPO (12/2011)	1 262.69 *s*	9.99 *s*	0.85 *s*	0.59 *s*	0.52 *s*	11.94 *s*	105.8
FMA-lite	19 991.92 *s*	9.50 *s*	0.49 *s*	6.43 *s*	0.20 *s*	16.62 *s*	1 202.9
Full-Galen (v1.1)	115.77 *s*	2.87 *s*	0.09 *s*	0.20 *s*	0.05 *s*	3.21 *s*	36.1
GO v1.1 (1986)	44.33 *s*	3.42 *s*	0.15 *s*	0.16 *s*	0.09 *s*	3.82 *s*	11.6
NCBI (v1.2)	49 227.86 *s*	73.72 *s*	1.42 *s*	0.87 *s*	1.19 *s*	77.22 *s*	637.5
RH-MeSH (08/2012)	6 921.62 *s*	15.24 *s*	1.31 *s*	0.63 *s*	0.66 *s*	17.84 *s*	388.0
Snomed CT (2010a)	4 538.65 *s*	14.47 *s*	0.48 *s*	0.31 *s*	0.32 *s*	15.58 *s*	291.3

The times for FaCT++ are the total times needed (2nd column), which includes loading the ontology, initialising the reasoner and computing the atoms (but not saving the atoms). For the hypergraph-based approach, we have listed the times needed for the OWL-API to load the ontology and to do some initialisation (3rd column), to build the axiom dependency hypergraph (4th column), to compute the strongly connected components of the graph (5th column), to compute the dependencies between the strongly connected components (6th column) and the total time needed (6th column). The last column shows for each ontology the speedup achieved for atomic decomposition using the hypergraph-based approach compared to FaCT++. On FMA-lite an over 1 000-fold speedup was realised.

FaCT++ improves upon the original algorithm for atomic decomposition [11] as described in [10]. However, the algorithm implemented in FaCT++ runs in time almost cubic (in particular, the computation of the transitive closure in Line 7 of Algorithm 4 in [10]). In contrast, the approach based on axiom dependency hypergraphs runs in time linear for the considered fragment of the ontologies.

5 Conclusion

We have suggested a hypergraph-based method for efficient atomic decomposition of $\mathcal{EL}$-ontologies consisting of primitive concept inclusions. We have introduced the notion of an axiom dependency hypergraph, in which, different to other hypergraph representations of ontologies, axioms are nodes that are connected in a way mimicking how axioms are included in a module by the locality-based module extraction algorithm.

Modularisation and atomic decomposition has been suggested as a means to understand the internal structure of ontologies. Axiom dependency hypergraphs are an explicit representation of these notions. A module can be extracted from the hypergraph by collecting the nodes reachable from the nodes that are determined by the input signature. Moreover, it is possible to directly apply standard off-the-shelf graph algorithms for computing the atoms of certain $\mathcal{EL}$-ontologies in linear time. For the atomic decomposition of FMA-lite, a staggering 1 000-fold speedup could be achieved compared to the reasoner FaCT++.

The disadvantage of the axiom dependency hypergraphs, as introduced in this paper, is their exponential size in the worst case for general concept inclusions. We have indicated that the blow-up in size can be avoided by omitting hyperedges while still preserving the mutual reachability relation between axioms. In this way, no more than quadratically many hyperedges are required for determining the atoms.

For future work, the idea of avoiding the exponential blow-up of the axiom dependency hypergraphs for general $\mathcal{EL}$-ontologies needs to be formalised, implemented and evaluated. It would also be interesting to extend the current approach to other locality notions such as $\top$- and $\bot\top^*$-locality and to richer languages such as $\mathcal{SROIQ}$.

References

1. X. Allamigeon. Strongly connected components of directed hypergraphs. *CoRR*, abs/1112.1444, 2011.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI-05*. Morgan-Kaufmann Publishers, 2005.
3. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, 2007.
4. G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(23):177–201, 1993.
5. B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: theory and practice. *JAIR*, 31:273–318, 2008.
6. R. Nortje, A. Britz, and T. Meyer. A normal form for hypergraph-based module extraction for SROIQ. In *Proc. of AOW 2012*, volume 969 of *CEUR Workshop Proceedings*, pages 40–51. CEUR-WS.org, 2012.
7. M. Sharir. A strong connectivity algorithm and its applications to data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981.
8. B. Suntisrivaraporn. *Polynomial time reasoning support for design and maintenance of large-scale biomedical ontologies*. PhD thesis, TU Dresden, Germany, 2009.
9. R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.
10. D. Tsarkov. Improved algorithms for module extraction and atomic decomposition. In *Proc. of DL'12*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
11. C. D. Vescovo, B. Parsia, U. Sattler, and T. Schneider. The modular structure of an ontology: Atomic decomposition. In *Proc. of IJCAI'11*, pages 2232–2237, 2011.

Towards a Unified Approach to Modular Ontology Development Using the Aspect-Oriented Paradigm

Ralph Schäfermeier and Adrian Paschke

Freie Universität Berlin,
Königin-Luise-Str. 24-26, 14195 Berlin, Germany
{schaef,paschke}@inf.fu-berlin.de
http://www.corporate-semantic-web.de/

Abstract. In this paper, we describe our ongoing work on the application of the Aspect-Oriented Programming paradigm to the problem of ontology modularization driven by overlapping modularization requirements. We examine commonalities between ontology modules and software aspects and propose an approach to applying the latter to the problem of a priori construction of modular ontologies and a posteriori ontology modularization.

Keywords: ontology modularization, aspect-oriented development, crosscutting concerns

1 Introduction

The majority of existing modularization approaches are specialized solutions, relying on semantic (cf., for example, [1–5]) or structural relatedness (e.g., [6–8]) and are only parametrizable to a limited degree. Parameters often reflect the internal operational mode of the modularization algorithm rather than requirements concerning the expected outcome of the modularization process from a user's point of view. Moreover, requirements may be related to different dimensions of the problem space. They can comprise functional requirements, i.e., requirements directly related to the problem domain or the task an ontology or ontology module is supposed to fulfill, and non-functional requirements, such as provenance information, multilingualism, or tractability of reasoning tasks.

The aspect-oriented programming paradigm allows for the separation of multidimensional requirements into dedicated software code modules (aspects), leading to better code modularity and therefore reusability. Using AOP, modules can be recombined (interwoven) either extensionally, by explicitly marking the points in the code where the execution flow should be diverted to a different module, or intensionally, by specifying a set of properties such code points are expected to have in common.

In this paper, we examine commonalities between ontology modules and software aspects and describe our ongoing work towards the application of the above mentioned principles of the AOP paradigm to ontologies. We argue that the latter enables (a) straightforward development of modular ontologies from scratch, and (b) flexible a-posteriori modularization, driven by user requirements.

2 The Aspect-Oriented Paradigm Applied to Ontologies

Aspect-oriented software programming (AOP) languages provide syntactic means for the separation of so called *cross-cutting concerns* in software code into dedicated modules. As defined by the ISO/IEC/IEEE standard 42010 of software architecture[1], *"concerns are those interests which pertain to the systems development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders"*. Cross-cutting concerns are concerns which emerge from requirements on different levels, concern the entire or a significant part of the system and are thus scattered across the system, diminishing code locality and hindering system evolution and reusability [9]. See Figure 1 for an explanatory example of cross-cutting concerns.

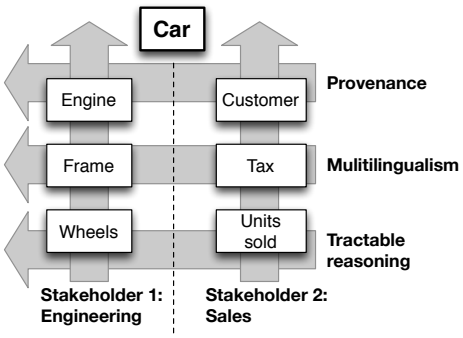


Fig. 1: Cross-cutting concerns by the example of a car ontology. Different stakeholders are interested in different aspects of the core concept (car). The different interests (concerns) reflect requirements formulated by each of the stakeholders. At the same time, stakeholder-independent requirements cross-cut the ontology. Each of theses requirements has a different dimension.

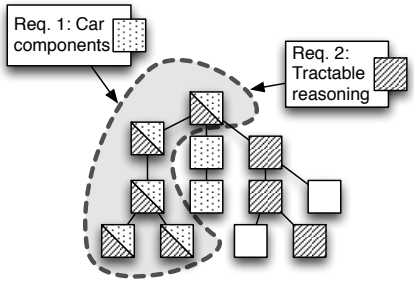


Fig. 2: Selection of an ontology module that satisfies two cross-cutting requirements: It should only contain concepts of the subdomain "car components" of the car domain ("Engineering" aspect, dotted), and it should only contain constructs that allow for tractable reasoning ("Tractability" aspect, dashed). The resulting module (grey) only contains those constructs that are concerned by both aspects.

In AOP terminology, the encapsulation of a single concern in a dedicated module is referred to as *aspect*. An aspect consists of two components: the actual implementation of the functionality, referred to as *advice*, and information about all points in the application's control flow at which the advice should be executed, referred to as *join points*. In this manner, we use the notion of aspects in order to relate ontological constructs to different requirements (see Figure 2).

Listing 1.1 shows an example of a concern "authentication" that cross-cuts with the actual business logic of a bank application and leads to tangled code. Listing 1.2 shows how the authentication concern is encapsulated in an aspect "Authentication". The authentication handling code has been centralized in the advice of the aspect.

[1] `http://www.iso-architecture.org/42010/`

Listing 1.1: Example of the cross-cutting concern "authentication" affecting different parts of the code.

```
withdraw (Account a, float amount) {
  AuthService as = getAuthService();
  if (!as.authenticated(a.user))
    as.authenticate(a.user);
  a.balance -= amount;
}
deposit(Account a, float amount) {
  AuthService as = getAuthService();
  if (!as.authenticated(a.user))
    as.authenticate(a.user);
  a.balance +=amount;
}
```

Listing 1.2: The concern "authentication" is encapsulated in an aspect "Authentication".

```
public Aspect Authentication() {
  AuthService as = getAuthService();
  if (!as.authenticated(a.user))
    as.authenticate(a.user);
}
@aspect Authentication
withdraw (Account a, float amount) {
  a.balance -= amount;
}
@aspect Authentication
deposit(Account a, float amount) {
  a.balance +=amount;
}
```

Note that this example demonstrates the explicit (extensional) variant of join points, in the form of tags assigned to each part of the code where an aspect is applicable. In order to achieve complete detangling of cross-cutting concerns, AOP introduces two principles: quantification and obliviousness.

2.1 Quantification

AOP allows for an intensional definition of join points by using quantified statements in the form

$$\forall m(p_1,\ldots,p_n) \in M : s(m(p_1,\ldots,p_n)) \rightarrow (m(p_1,\ldots,p_n) \rightarrow a(p_1,\ldots,p_n)),$$

where M is the set of all methods defined within the software product, s a predicate specifying the join point properties, $m(p_1,\ldots,p_n) \in M$ a method adhering to the signature $m(p_1,\ldots,p_n)$, and $a(p_1,\ldots,p_n)$ the execution of the advice with all the parameters of each method, respectively [10]. The set of all join points defined by s is called a *pointcut*.

In the case of the authentication example, an instantiation of this formula would be:

$$\forall m(p_1,\ldots,p_n) \in M : sig(m(p_1,\ldots,p_n)) = m(Account\ acc, float\ amount) \\ \rightarrow (m(acc, amount) \rightarrow Authentication(acc, amount)).$$

In order to select ontology axioms in the same fashion, a means of quantification over such axioms is necessary.

$$\forall ax(p_1,\ldots p_n) \in \mathcal{O} : s(ax(p_1,\ldots p_n)) \rightarrow (ax(p_1,\ldots p_n) \rightarrow a(ax(p_1,\ldots p_n))),$$

with $\mathcal{O}$ being an ontology, $ax(p_1,\ldots p_n)$ axioms (in the form of n-ary predicates the domain of which is the union of the vocabulary of the ontology language

in question and the vocabulary of the problem domain, such as class, property and individual names), and $a(ax(p_1, \ldots p_n))$ a function that applies the aspect to $ax(p_1, \ldots p_n)$, whereupon we define the application of an aspect to an axiom as the inclusion of that particular axiom in the module represented by the aspect. In this manner, the aspect "tractability" from the example scenario could be implemented by building a query that selects all axioms which are compatible with the OWL-EL profile.

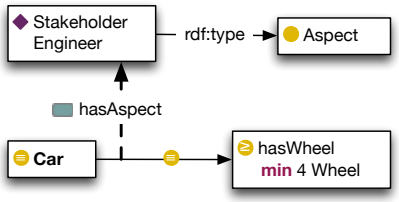


Fig. 3: Extensional join point definition using an OWL annotation.

As mentioned in section 1, join points can also be specified extensionally, for example, by manually annotating axioms which cover a particular aspect with a dedicated OWL Annotation (see Figure 3). This is of particular use for a priori modular ontology development if used, e. g., by an ontology editor with switchable contexts, each context representing an aspect. A user could then extensionally define an aspect-oriented ontology module by switching aspects.

2.2 Obliviousness and Harmlessness

The fact that in aspect-oriented software systems control flow is handed over from the main module to the aspects, making the main module (and any other modules) unaware of the (quantified or extensionally specified) assertions made about it by external aspects that might possibly be applied to it, is termed *obliviousness* [9]. The practical consequence of obliviousness is that the developer of a module is not required to have knowledge about or spend additional effort in anticipation of a possible application of an aspect to her module.

Danters et al. allude to the problem that obliviousness is only guaranteed on a syntactic level while external aspects have in fact the potential to alter the semantics of the target module, making them potentially dangerous [11]. They propose the adaption of the *harmlessness* property for aspect-oriented systems, defining a *harmless aspect* as an aspect which, when applied to a target module, does not alter the semantics of the target.

While it is obvious that the obliviousness property naturally holds for ontology modules, the harmlessness property does not. Nevertheless, it is agreed upon in the recent literature that it is desirable if an ontology module is uninvasive, i.e., its addition to an ontology has no side effects. Whether uninvasiveness of a module applied by the means of an aspect is a required feature or not depends on the specific use case.

Grau et al. [12] as well as Konev et al. [13] propose the notion of conservative extensions which guarantee that a module added to an ontology does not alter the meaning of the original ontology:

Let $\mathcal{O}_1 \subseteq \mathcal{O}$ ontologies, S a signature and L a logic. $\mathcal{O}$ is a conservative extension of $\mathcal{O}_1$ wrt. L, if for every axiom ax with $sig(ax) \subseteq \mathcal{S}$: $\mathcal{O} \models ax$ iff $\mathcal{O}_1 \models ax$.

In the same vein, we define a harmless aspect of an ontology $\mathcal{O}$ as an aspect that yields the selection of a module $\mathcal{O}_1$ that is the conservative extension of $\mathcal{O}$ with respect to L.

It has been noted that determining whether a module is a conservative extension of an ontology is a highly complex problem and even undecidable for expressive ontologies [12, 13]. However, semantic locality is a sufficient condition for a conservative extension [4], and [14] suggests that the less complex syntactic locality constitutes a practically acceptable approximation.

3 Conclusion and Future Work

In this paper, we pointed out that ontology modularization and aspect-oriented programming share interesting commonalities and that the aspect-oriented paradigm can be applied to a priori modular ontology development as well as a posteriori module extraction. The next step will consist in providing a proof-of-concept system that dynamically interweaves aspects defined in the above manner.

Further work is necessary in order to achieve a functional meta description of ontology axioms for the purpose of pointcut definition. The formalism described in section 2.1 works in terms of meta predicates with the domain consisting of vocabulary of the ontology language, reifing axioms contained in the ontology.

The research question raised in this paper is how the application of the aspect-oriented paradigm affects the quality of ontology modularizations. Our hypothesis is that aspect-oriented ontology development yields useful ontology modules wrt. to cross-cutting modularization requirements, such as dynamic access, understandability, maintenance, and re-use. We expect that the intensional specification of ontology modules with pointcuts adds dynamicity and flexibility to modular development, making it easier to evolve modular ontologies in situations where evolution implies modularization requirement changes.

To evaluate our approach and test our hypothesis, we will apply the approach to different modularization use-cases in the context of ontology development projects. Aspects considered in these use cases will comprise project affiliation, temporal attribution, workflow affiliation, re-use, and module understandability. We then use quality metrics in order to assess the quality of the modularizations gained using our approach and compare it with existing approaches.

Acknowledgements

This work has been partially supported by the "InnoProfile-Transfer Corporate Smart Content" project funded by the German Federal Ministry of Education and Research (BMBF) and the BMBF Innovation Initiative for the New German Länder - Entrepreneurial Regions.

References

1. Cuenca Grau, B., Parsia, B., Sirin, E.: Combining OWL ontologies using $\mathcal{E}$-Connections. Web Semantics: Science, Services and Agents on the World Wide

Web **4**(1) (January 2006) 40–59
2. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic Modularity and Module Extraction in Description Logics. In: Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence, Amsterdam, The Netherlands, The Netherlands, IOS Press (2008) 55–59
3. Suntisrivaraporn, B.: Module Extraction and Incremental Classification: A Pragmatic Approach for $\mathcal{EL}^+$ Ontologies. In Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M., eds.: The Semantic Web: Research and Applications. Number 5021 in Lecture Notes in Computer Science. Springer Berlin Heidelberg (January 2008) 230–244
4. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Extracting Modules from Ontologies: A Logic-Based Approach. [15] 159–186 DOI: 10.1007/978-3-642-01907-4.
5. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to DL-Lite. Artificial Intelligence **174**(15) (October 2010) 1093–1141
6. d'Aquin, M., Doran, P., Motta, E., Tamma, V.A.M.: Towards a parametric ontology modularization framework based on graph transformation. In Grau, B.C., Honavar, V., Schlicht, A., Wolter, F., eds.: Proceedings of the 2nd International Workshop on Modular Ontologies, WoMO 2007. Volume 315 of CEUR Workshop Proceedings., CEUR-WS.org (2007)
7. Schlicht, A., Stuckenschmidt, H.: A Flexible Partitioning Tool for Large Ontologies. In: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01. WI-IAT '08, Washington, DC, USA, IEEE Computer Society (2008) 482—488
8. Coskun, G., Rothe, M., Teymourian, K., Paschke, A.: Applying community detection algorithms on ontologies for indentifying concept groups. In: Proceedings of the 5th International Workshop on Modular Ontologies, Ljubljana, Slovenia (September 2011)
9. Filman, R., Friedman, D.: Aspect-Oriented Programming Is Quantification and Obliviousness. Workshop on Advanced Separation of Concerns, OOPSLA (2000)
10. Steimann, F.: Domain Models Are Aspect Free. In Briand, L., Williams, C., eds.: Model Driven Engineering Languages and Systems. Number 3713 in Lecture Notes in Computer Science. Springer Berlin Heidelberg (January 2005) 171–185
11. Dantas, D.S., Walker, D.: Harmless Advice. In: Conference record of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages. POPL '06, New York, NY, USA, ACM (2006) 383–396
12. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular Reuse of Ontologies: Theory and Practice. Journal of Artificial Intelligence Research **31** (February 2008) 273–318 ACM ID: 1622664.
13. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal Properties of Modularisation. [15] 25–66 DOI: 10.1007/978-3-642-01907-4.
14. Del Vescovo, C., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Tsarkov, D.: Syntactic vs. Semantic Locality: How Good Is a Cheap Approximation? In Schneider, T., Walther, D., eds.: Workshop on Modular Ontologies (WoMO) 2012. (2012) 40–50
15. Stuckenschmidt, H., Parent, C., Spaccapietra, S., eds.: Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization. Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2009) DOI: 10.1007/978-3-642-01907-4.