Proceedings of the 6th International

Workshop on Modular Ontologies (WoMO) 2012

held at Graz, Austria, on July 24, 2012
as a satellite event of ICBO/FOIS 2012

Workshop chairs and proceedings editors:

Thomas Schneider, University of Bremen, Germany
Dirk Walther, Universidad Politécnica de Madrid, Spain

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 2012 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).

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

Organizers of the previous editions and editors of the proceedings were Diego Calvanese (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; 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; Frank Wolter (Liverpool) – 2007, 2008

This volume contains the papers presented at the 6th International Workshop on Modular Ontologies (WoMO 2012) held on July 24, 2012 in Graz, as a satellite event of the joint ICBO/FOIS conferences. There were nine submissions. Each submission was reviewed by three program committee members. The committee decided to accept eight papers for long or short presentations. The program also included two invited talks:

– Thomas Eiter (Vienna University of Technology, Austria)
Distribution and Modularity in Nonmonotonic Logic Programming

– Luciano Serafini (Fondazione Bruno Kessler, Trento, Italy)
Multi Context Logics: a Formal Framework for Structuring Knowledge

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 the ICBO/FOIS conferences for hosting the WoMO workshop, the IAOA for complimentary registration of invited speakers and one organizer, and the EasyChair developers for greatly simplifying the work of the program committee.

July 12, 2012 Thomas Schneider
Bremen and Madrid Dirk Walther

Table of Contents

Summaries of Invited Talks

Regular Papers

Short Papers

Program Committee

Kenneth Baclawski	VIStology, Inc., Framingham, MA, USA
Stefano Borgo	CNR, Trento, Italy
Oscar Corcho	Universidad Politécnica de Madrid, Spain
Bernardo Cuenca Grau	University of Oxford, UK
Mike Dean	Raytheon BBN Technologies, Ann Arbor, MI, US
Silvio Ghilardi	Universita degli Studi di Milano, Italy
Michael Gruninger	University of Toronto, Canada
Janna Hastings	Europ. Bioinformatics Institute, Cambridge, UK
Robert Hoehndorf	University of Cambridge, UK
Boris Konev	The University of Liverpool, UK
Roman Kontchakov	Birkbeck College, London, UK
Thomas Meyer	Meraka Institute, CSIR, Pretoria, South Africa
Till Mossakowski	DFKI Lab, Bremen, Germany
Immanuel Normann	Birkbeck College, London, UK
Leo Obrst	MITRE, McLean, VA, USA
Adrian Paschke	Freie Universität Berlin, Germany
David Perez-Rey	Universidad Politécnica de Madrid, Spain
Ulrike Sattler	University of Manchester, UK
Thomas Schneider (co-chair)	University of Bremen, Germany
Luciano Serafini	FBK-IRST, Trento, Italy
Dirk Walther (co-chair)	Universidad Politécnica de Madrid, Spain

Additional Reviewer

Riku Nortje

Author Index

Distribution and Modularity in Nonmonotonic Logic Programming

Thomas Eiter

Knowledge-Based Systems Group
Institute of Information Systems,
Vienna University of Technology
eiter@kr.tuwien.ac.at

In the recent years, there has been a trend towards considering computation in a distributed setting, due to the fact that increasingly not only data is linked via media such as the internet, but also computational entities which process and exchange data and knowledge. This leads to the formation of (possibly complex) systems of interlinked entities, based on possibly heterogenous formalisms, posing challenging issues on semantics and computation. The concept of modularity, which in computer science and engineering is a key to structured program development, naturally links to this as a tool for defining semantics of distributed systems, and has been widely studied, e.g., in the area of ontologies. In line with the general development, distribution and modularity have been also been receiving increased attention in logic programming, at several levels of language expressiveness, from distributed (plain) datalog to advanced nonmonotonic logic programming semantics.

In this talk, we shall address the issue of distribution and modularity for logic programming under the answer set semantics, which is one of the most widely used semantics for nonmontonic logic programs do date and at the heart of the Answer Set Programming paradigm for declarative problem solving. It appeared that the issue of modularity for answer set semantics is nontrivial, due to its nonmonotonicity. For the same reason, also the issue of efficient distributed evaluation, assuming a reasonable behavior of the semantics for a program composed of distributed modules, is a challenging problem. We shall discuss these issues, pointing out that modularity and distribution admit different solutions for semantics, depending on the underlying view of a system of logic programs. We then illustrate this view on particular formalisms that have been developed at the Vienna University of Technology in the last years, including modular nonmonotonic logic programs (Modular ASP) and nonmonotonic multi-context systems (MCS). For these formalisms, various semantics have been developed, as well as experimental prototype implementations that take local or distributed evaluation into account, adopting different realization schemes. While considerable progress has been achieved, further work is needed to arrive at highly efficient solvers.

This work is a joint effort with Minh Dao-Tran, Michael Fink, Thomas Krennwallner and Tri Kurniawan Wijaya, supported by the project P20841 “Modular HEX-Programs” of the Austrian Science Fund (FWF).

Multi context logics: a formal framework for structuring knowledge

Luciano Serafini

Fondazione Bruno Kessler
via Sommarive 18, I-38123, Trento, Italy
serafini@fbk.eu

Multi context logics (MCL) is a formalism that allow to integrate multiple logical theories (contexts) in a more complex structure called multi context system. In the past 20 years multi context logics have been developed for contexts in propositional logics, first order logics, description logics and temporal logic. The two principles MCL are *locality* and *compatibility*. The principle of locality states that a context axiomatizes in a logical theory a portion of the world, and that every statement entailed by such a theory is intended to hold within such a portion of the world. The principle of compatibility instead states that, since different contexts can describe overlapping portions of world, the theories they contain must be constrained so that they describe compatible situations. Following these two principles, the formal semantics of an MCL is the result of a suitable composition of the semantics associated to each single context. This takes the name of *Local models semantics*. The effects of the two principles above on the inference engines that can be defined on a multi contextual knowledge base are the following: Locality principle implies that inference rules applied to knowledge inside a context (aka, local inference rules) allow to infer local truths; Compatibility principle instead implies that certain facts in a context can be inferred on the base of other facts present on other compatible contexts. This information propagation is formalized via a special type of inter contextual inference rules called *bridge rules*

In general terms, a multi context logic is defined on a family of logical languages $\{L_i\}_{i\in I}$ where each L_i is used to specify what holds in the i-th context. The set I of context indexes (aka context names) can be either a simple set, or a set equipped with some algebraic structure, like total or partial order, and operations on context indices. The relations and functions defined on I can be used to specify the organization of contexts in terms of an algebraic structure. For instance a partial order $\prec$ no I, can be used to represent that a context is wider (more general) than another context, e.g., $\mathsf{football} \prec \mathsf{sport}$, means that the context of sport is more specific that the context of football. To represent what is true in a scene in different time stamps, we can enrich I with a total order $\prec$ so that $\mathsf{CV\_Luciano\_2010} \prec \mathsf{CV\_Luciano\_2011}$ represents the fact that the context describing Luciano's curriculum vitæat 2010 precedes the context describing his CV at 2011.

A *model* for a multi context logic $\{L_i\}_{i\in I}$ is a pair $\langle M_I, C\rangle$ composed of a family of local models $M_I = \{M_i\}_{i\in I}$, where each M_i is a model of L_i, and a *compatibility relation* C among the local models. The formal structure of C

can vary depending on the type of local models and the type of constraints it is necessary to impose on the local models of different contexts. For this reason we don't give a general definition of C, which will be completely defined for each specific multi context logic. Satisfiability of formulas in L_i is defined w.r.t1. the local models. Namely If ϕ is a formula of the language L_i, then the a multi context model satisfies $i : \phi$ iff $M_i \models_i \phi$, where $\models_i$ is the satisfiability relation associated to the local logic L_i.

A *Multi context theory* in a multi context logic $L_I = \{L_i\}_{i \in I}$ is a family of theories $\{T_i\}_{i \in I}$, where each T_i is a set of statements in the logics L_i, and a set BR of *bridge rules*. Intuitively each T_i axiomatizes the constraints on the local models M_i, while the bridge rules BR axiomatizes the compatibility relations. Bridge rules are cross logical axioms and their syntactic form depends on the local logics, so as in the case of the compatibility relation their syntactic depends on the syntax of each L_i.

In my invited talk, I will go through the many possible examples of multi context logics, starting from the simplest one, the propositional multi context logics, going through hierarchical meta logics, multi context logics for beliefs and propositional attitudes, non-monotonic propositional context logics, distributed first order logics, distributed description logics, and logics for semantic import and contextualized knowledge repository for the semantic web.

Structure Formation to Modularize Ontologies

Serge Autexier and Dieter Hutter *

German Research Center for Artificial Intelligence
Bibliotheksstr. 1, 28359 Bremen, Germany
{autexier|hutter}@dfki.de

Abstract. It has been well recognized that the structuring of logic-based databases like formal ontologies is an important prerequisite to allow for an efficient reasoning on such databases. Development graphs have been introduced as a formal basis to represent and reason on structured specifications. In this paper we present an initial methodology and a formal calculus to transform unstructured specifications into structured ones. The calculus rules operate on development graphs allowing one to separate specifications coalesced in one theory into a concisely structured graph.

1 Introduction

It has been long recognized that the modularity of specifications is an indispensable prerequisite for an efficient reasoning in complex domains. Algebraic specification techniques provide frameworks for structuring complex specifications and the authors introduced the notion of an development graph [4, 1, 6] as a technical means to work with and reason about such structured specifications. Typically ontologies are large and, even if structured, bear the problem of inconsistencies as any large set of axioms. For instance, the SUMO ontology [7] turned out to be inconsistent [10]. Recently Kurz and Mossakowski presented an approach [5] to prove the consistency of an ontology in a modular way using the (structured) architectural specification in CASL and provide mechanisms to compose the models of the individual components to a global one. Furthermore, there has been work [9] to (re-)structure ontologies following locality criteria into modules which cover all aspects about specific concepts. However, since ontology languages have simple imports without renaming, duplicated sub-ontologies that are for instance equal up to renaming remain hidden, thus making the ontologies unnecessarily large, not only from a modeling point of view, but also from a verification point of view as the same derived properties must be derived over an over again.

In this paper we present further steps towards the support for structure formation in large specifications that additionally allows to factorize equivalent sub-specification, i.e. sub-ontologies. The idea is to provide a calculus and a corresponding methodology to crystallize intrinsic structures hidden in a specification and represent them explicitly in terms of development graphs. We start

* This work was funded by the German Federal Ministry of Education and Research under grants 01 IS 11013B and 01 IW 10002 (projects SIMPLE and SHIP)

with a trivial development graph consisting of a single node that contains the entire specification. Step by step, the specification is split to different nodes in the development graph resulting in an increasingly richer graph. On the opposite, common concepts that are scattered in different specifications are identified and unified in a common theory (i.e. node).

2 Prerequisites

We base our framework on the notions of development graphs to specify and reason about structured specifications. Development graphs $\mathcal{D}$ are acyclic, directed graphs $\langle \mathcal{N}, \mathcal{L} \rangle$, the nodes $\mathcal{N}$ denote individual theories and the links $\mathcal{L}$ indicate theory inclusions with respect to signature morphisms attached to the links.

This approach is based on the notion of institutions [3] ($\mathbf{Sign}_{\mathcal{I}}, \mathbf{Sen}_{\mathcal{I}}, \mathbf{Mod}_{\mathcal{I}}$, $\models_{\mathcal{I},\Sigma}$), where (i) **Sign** is a category of *signatures*, (ii) $\mathbf{Sen}\colon \mathbf{Sign} \longrightarrow \mathbf{Set}$ is a functor giving the set of *sentences* $\mathbf{Sen}(\Sigma)$ over each signature Σ, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the sentence translation function $\mathbf{Sen}(\sigma)\colon \mathbf{Sen}(\Sigma) \longrightarrow \mathbf{Sen}(\Sigma')$, where often $\mathbf{Sen}(\sigma)(\varphi)$ is written as $\sigma(\varphi)$, (iii) $\mathbf{Mod}\colon \mathbf{Sign}^{op} \longrightarrow \mathcal{CAT}$ is a functor giving the category of *models* over a given signature, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the *reduct functor* $\mathbf{Mod}(\sigma)\colon \mathbf{Mod}(\Sigma') \longrightarrow \mathbf{Mod}(\Sigma)$, where often $\mathbf{Mod}(\sigma)(M')$ is written as $M'|_\sigma$, (iv) $\models_\Sigma \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$ for each $\Sigma \in |\mathbf{Sign}|$ is a satisfaction relation,[1] such that for each $\sigma\colon \Sigma \longrightarrow \Sigma'$ in **Sign**, $M' \models_{\Sigma'} \sigma(\varphi) \Leftrightarrow M'|_\sigma \models_\Sigma \varphi$ holds for each $M' \in \mathbf{Mod}(\Sigma')$ and $\varphi \in \mathbf{Sen}(\Sigma)$ (*satisfaction condition*).

However, the abstractness of the signatures in the definition of institution (they are just a category) makes it difficult to cope with modularization of signatures. Hence, we equip institutions with an additional structure such that signatures behave more set-like. *Institutions with pre-signatures* [2] are defined as institutions equipped with an embedding $|\_| : \mathbf{Sign} \to \mathbf{Set}$, the *symbol functor*, and a map $sym\colon \bigcup_{\Sigma \in |\mathbf{Sign}|} \mathbf{Sen}(\Sigma) \to |\mathbf{Set}|$, such that $\varphi \in \mathbf{Sen}(\Sigma)$ iff $sym(\varphi) \subseteq |\Sigma|$ for all $\varphi \in \bigcup_{\Sigma \in |\mathbf{Sign}|} \mathbf{Sen}(\Sigma)$. The map *sym* gives the set of symbols used in a sentence, and sentences are uniform in the sense that a well-formed sentence is well-formed over a certain signature iff its symbols belong to that signature. Pre-signatures are sets and pre-signature morphisms $\bar{\sigma}$ mapping pre-signatures are related to corresponding signature morphisms σ.

Given an $\mathcal{I}$-institution with pre-signatures, each node $N \in \mathcal{N}$ of the graph is a tuple (sig^N, ax^N, lem^N) such that sig^N is a $\mathcal{I}$-pre-signature called the *local signature* of N, ax^N a set of $\mathcal{I}$-sentences called the *local axioms* of N, and lem^N a set of $\mathcal{I}$-sentences called the *local lemmas* of N. $\mathcal{L}$ is a set of global definition links $M \xRightarrow{\bar{\sigma}} N$. Such a link imports the mapped theory of M (by the pre-signature $\bar{\sigma}$) as part of the theory of N. Thus we obtain corresponding notions of global (pre-)signatures, axioms and lemmata that are defined inductively as follows:

[1] $|C|$ is the class of objects of a category C.

1. $Sig_{\mathcal{D}}(N) = sig^N \cup \bigcup_{M \xrightarrow{\bar{\sigma}} N \in S} \bar{\sigma}(Sig_{\mathcal{D}}(M))$
2. $Ax_{\mathcal{D}}(N) = ax^N \cup \bigcup_{M \xrightarrow{\bar{\sigma}} N \in \mathcal{D}} \sigma(Ax_{\mathcal{D}}(M))$
3. $Lem_{\mathcal{D}}(N) = lem^N \cup \bigcup_{M \xrightarrow{\bar{\sigma}} N \in \mathcal{D}} \sigma(Lem_{\mathcal{D}}(M))$

A node $N \in \mathcal{N}$ is globally reachable from a node $M \in \mathcal{N}$ via a pre-signature morphism $\bar{\sigma}$, $\mathcal{D} \vdash M \overset{\bar{\sigma}}{\rightarrowtail} N$ for short, iff 1. either $M = N$ and $\bar{\sigma} = id$ is the identity pre-signature morphism, or 2. $M \xrightarrow{\bar{\sigma}'} K \in \mathcal{L}$, and $\mathcal{D} \vdash K \overset{\bar{\sigma}''}{\rightarrowtail} N$, with $\bar{\sigma} = \bar{\sigma}'' \circ \bar{\sigma}'$.

The *maximal nodes* (root nodes) $\lceil \mathcal{D} \rceil$ of a graph $\mathcal{D}$ are all nodes without outgoing links. $Dom_{\mathcal{D}}(N) := Sig_{\mathcal{D}}(N) \cup Ax_{\mathcal{D}}(N) \cup Lem_{\mathcal{D}}(N)$ is the set of all signature symbols, axioms and lemmata visible in a node N. The *local domain* of N, $dom^N := sig^N \cup ax^N \cup lem^N$ is the set of all local signature symbols, axioms and lemmata of N. The *imported domain* $Imports_{\mathcal{D}}(N)$ of N in $\mathcal{D}$ is the set of all signature symbols, axioms and lemmas imported via incoming definition links. $Dom_{\mathcal{D}} = \bigcup_{N \in \mathcal{N}} Dom_{\mathcal{D}}(N)$ is the set of all signature symbols, axioms and lemmata occurring in $\mathcal{D}$. Analogously we define $Sig_{\mathcal{D}}$, $Ax_{\mathcal{D}}$, $Lem_{\mathcal{D}}$, and $Ass_{\mathcal{D}}$. $Dom_{\lceil \mathcal{D} \rceil} = \bigcup_{N \in \lceil \mathcal{D} \rceil} Dom_{\mathcal{D}}(N)$ is the set of all signature symbols, axioms and lemmata occurring in the maximal nodes of $\mathcal{D}$.

A node N has a well-formed signature iff $Sig_{\mathcal{D}}(N)$ is a valid $\mathcal{I}^N$-signature. A development graph has a well-formed signature iff all its nodes have well-formed signatures. $Sig^{loc}_{\mathcal{D}}(M) := \langle sig^M \cup sym(ax^M) \cup sym(lem^M) \rangle_{Sig_{\mathcal{D}}(M)}$ is the *local signature* of N. A node N is well-formed iff it has a well-formed signature $Sig_{\mathcal{D}}(N)$ and $Ax_{\mathcal{D}}(N), Lem_{\mathcal{D}}(N) \subseteq \mathbf{Sen}(Sig_{\mathcal{D}}(N))$. A development graph is well-formed, if all its nodes are well-formed.

Given a node $N \in \mathcal{N}$ with well-formed signature, its associated class $\mathbf{Mod}^{\mathcal{D}}(N)$ of models (or N-models for short) consists of those $Sig_{\mathcal{D}}(N)$-models n for which (i) n satisfies the local axioms ax^N, and (ii) for each $K \xrightarrow{\bar{\sigma}} N \in \mathcal{S}$, $n|_\sigma$ is a K-model. In the following we denote the class of Σ-models that fulfill the Σ-sentences Ψ by $\mathbf{Mod}_\Sigma(\Psi)$. A well-formed development graph $\mathcal{D} := \langle \mathcal{N}, \mathcal{L} \rangle$ is *valid* iff for all nodes $N \in \mathcal{N}$ $\mathbf{Mod}^{\mathcal{D}}(N) \models lem^N$.

Given a signature Σ and $Ax, Lem \subseteq \mathbf{Sen}(\Sigma)$, a *support mapping Supp for Ax and Lem* assigns each lemma $\varphi \in Lem$ a subset $H \subseteq Ax \cup Lem$ such that (i) $\mathbf{Mod}_{\langle sym(H) \cup sym(\varphi) \rangle_\Sigma}(H) \models \varphi$ (ii) The relation $\sqsubset \subseteq (Ax \cup Lem) \times Lem$ with $\Phi \sqsubset \varphi \Leftrightarrow (\Phi \in Supp(\varphi) \vee \exists \psi . \Phi \in Supp(\psi) \wedge \psi \sqsubset \varphi)$ is a well-founded strict partial order. If $\mathcal{D}$ is a development graph, then a support mapping *Supp* is a *support mapping for* $\mathcal{D}$ iff for all $N \in \mathcal{D}$ *Supp* is a support mapping for $Ax_{\mathcal{D}}(N)$ and $Lem_{\mathcal{D}}(N)$.

3 Development Graphs for Structure Formation

As a first step towards structure formation we will formalize requirements on development graphs that reflect our intuition of an appropriate structuring for

(formal) ontologies in particular (and formal specifications in general) in the following principles.

The first principle is *semantic appropriateness*, saying that the structure of the development graph should be a syntactical reflection of the relations between the various concepts in our ontology. This means that different concepts are located in different nodes of the graph and the links of the graph reflect the logical relations between these concepts. The second principle is *closure* saying, for instance, that deduced knowledge should be located close to the concepts guaranteeing the proofs. Also the concept defined by the theory of an individual node of a development graph should have a meaning of its own and provide some source of deduced knowledge. The third principle is *minimality* saying that each concept (or part of it) is only represented once in the graph. When splitting a monolithic theory into different concepts common foundations for various concepts should be (syntactically) shared between them by being located at a unique node of the graph.

In the following we translate these principles into syntactical criteria on development graphs and also into rules to transform and refactor development graphs. In a first step we formalize technical requirements to enforce the minimality-principle in terms of development graphs. Technically, we demand that each signature symbol, each axiom and each lemma has a unique location in the development graph. When we enrich a development graph with more structure we forbid to have multiple copies of the same definition in different nodes. We therefore require that we can identify for a given signature entry, axiom or lemma a *minimal theory* in a development graph and that this minimal theory is unique. We define:

Definition 1 (Providing Nodes). *Let $\langle \mathcal{N}, \mathcal{L} \rangle$ be a development graph. An entity e is* provided *in $N \in \mathcal{N}$ iff $e \in Dom_{\langle \mathcal{N},\mathcal{L} \rangle}(N)$ and $\forall M \xrightarrow{\bar{\sigma}} N.\ e \notin Dom_{\langle \mathcal{N},\mathcal{L} \rangle}(M)$. Furthermore,*

1. *e is* locally *provided in N iff additionally $e \in dom^N$ holds.*
2. *e is provided* by a link $l : M \xrightarrow{\bar{\sigma}} N$ *if not locally provided in N and $\exists e' \in Dom_{\langle \mathcal{N},\mathcal{L} \rangle}(M).\ \sigma(e') = e$. In this case we say that l provides e from e'. e is* exclusively *provided by l iff e is not provided by any other link $l' \in \mathcal{L}$.*

Finally, the closure-principle demands that there are no spurious nodes in the graph that do not contribute anything new to a concept. We combine these requirements into the notion of location mappings:

Definition 2 (Location Mappings). *Let $\mathcal{D} = \langle \mathcal{N}, \mathcal{L} \rangle$ be a development graph. A mapping $loc_{\mathcal{D}} : Dom_{\mathcal{D}} \to \mathcal{N}$ is a* location *mapping for $\mathcal{D}$ iff*

1. *$loc_{\mathcal{D}}$ is surjective (closure)*
2. *$\forall N \in \mathcal{N}.\ \forall e \in dom^N.\ loc_{\mathcal{D}}(e) = N$*
3. *$\forall e \in Dom_{\mathcal{D}}.\ loc_{\mathcal{D}}(e)$ is the only node providing e (minimality)*

Furthermore, for a given $loc_{\mathcal{D}}$ we define $loc_{\mathcal{D}}^{-1} : \mathcal{N} \to 2^{Dom_{\mathcal{D}}}$ by $loc_{\mathcal{D}}^{-1}(N) := \{e \in Dom_{\mathcal{D}} | loc_{\mathcal{D}}(e) = N\}$. We will write loc and loc^{-1} instead of $loc_{\mathcal{D}}$ and $loc_{\mathcal{D}}^{-1}$ if $\mathcal{D}$ is clear from the context.

Based on the notion of location mappings we formalize our intuition of a *structuring*. The idea is that the notion of being a structuring constitutes the invariant of the structure formation process and guarantees both, requirements imposed by the minimality-principle as well as basic conditions on a development graph to reflect a given formal specification or ontology.

Definition 3 (Structuring). *Let $\mathcal{D} = \langle \mathcal{N}, \mathcal{L} \rangle$ be a valid development graph, $loc : Dom_{\mathcal{D}} \to \mathcal{N}$, $\Sigma \in |\mathbf{Sign}|$, $Ax, Lem \subseteq \mathbf{Sen}(\Sigma)$ and $Supp$ be a support mapping for $\mathcal{D}$. Then $(\mathcal{D}, loc, Supp)$ is a* structuring *of (Σ, Ax, Lem) iff*

1. *loc is a location mapping for $\mathcal{D}$.*
2. *let $Dom_{\lceil \mathcal{D} \rceil} = \Sigma' \cup Ax' \cup Lem'$ then $\Sigma = \Sigma'$, $Ax = Ax'$ and $Lem \subseteq Lem'$.*
3. *$\forall \phi \in Lem_{\mathcal{D}} \,.\, \forall \psi \in Supp(\phi).\ \exists \bar{\sigma}.\ loc(\psi) \supset\!\!\stackrel{\bar{\sigma}}{\Longrightarrow} loc(\phi) \wedge \bar{\sigma}(\psi) = \psi$*

4 Refactoring Rules

In the following we present the transformation rules on development graphs that transform structurings again into structurings. Using these rules we are able to structure the initially trivial development graph consisting of exactly one node that comprises all given concepts step by step. This initial development graph consisting of exactly one node satisfies the condition of a structuring provided that we have an appropriate support mapping at hand.

We define four types of structuring-invariant transformations: (i) horizontal splitting and merging of development graph nodes, (ii) vertical splitting and merging of development graph nodes, (iii) factorization and multiplication of development graph nodes, and (iv) removal and insertion of specific links. Splitting and merging as well as factorization and multiplication are dual operations. For lack of space and because we are mainly interested in rules increasing the structure of a development graph we will omit the formal specification of the merging and multiplication rules here.

We illustrate our rules with the help of a running example in mathematics. We start from a flat theory specifying a Ring over two operations $+$ and $\times$. A structure $(R, +, \times)$ is a Ring, if $(R, \times)$ is an abelian group, $(R, +)$ a monoid and $\times$ distributes over $+$. Furthermore, an abelian group is a monoid for which additionally commutativity holds and inverse exists. The initial development graph consists of a single node (without any links) containing all the symbol definitions, axioms and theorems of the example.

Vertical Split. The first refactoring rule aims at the split of specifications in different theories. In terms of the development graph a node is replaced by two nodes one of them importing the other; each of them contains a distinct part from a partitioning of the specification of the original node. While all outgoing links start at the top node, we are free to reallocate incoming links to either node. To formalize this rule we need constraints on how to split a specification in different chunks such that local lemmata are always located in a node which provides also the necessary axioms and lemmata to prove it.

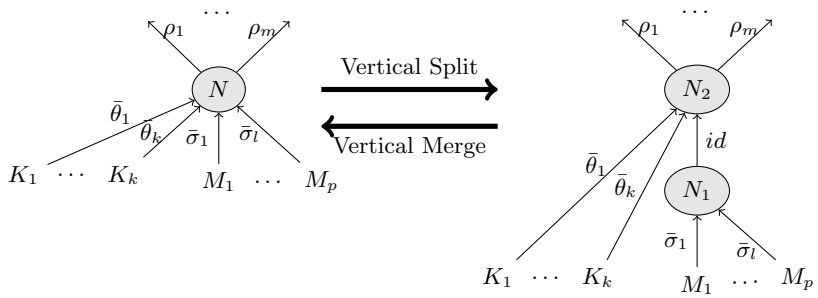

Fig. 1. Vertical Split and Merge

Definition 4. *Let $\mathcal{S} = (\mathcal{D}, loc, Supp)$ be a structuring of (Σ, Ax, Lem) and $N \in \mathcal{N}_\mathcal{D}$. A partitioning $\mathcal{P}$ for N is a set $\{N_1, \ldots, N_k\}$ with $k > 1$ such that 1. $sig^N = sig^{N_1} \uplus \ldots \uplus sig^{N_k}$, $ax^N = ax^{N_1} \uplus \ldots \uplus ax^{N_k}$, $lem^N = lem^{N_1} \uplus \ldots \uplus lem^{N_k}$ 2. $sig^{N_i} \cup ax^{N_i} \cup lem^{N_i} \neq \emptyset$ for $i = 1, \ldots, k$. A node $N_i \in \mathcal{P}$ is* lemma independent *iff $Supp(\psi) \cap (ax^N \cup lem^N) \subseteq (ax^{N_i} \cup lem^{N_i})$ for all $\psi \in lem^{N_i}$.*

Definition 5 (Vertical Split). *Let $\mathcal{S} = (\langle \mathcal{N}, \mathcal{L} \rangle, loc, Supp)$ be a* structuring *of (Σ, Ax, Lem) and $\mathcal{P} = \{N_1, N_2\}$ be a partitioning for some $N \in \mathcal{N}$ such that N_1 is lemma independent. Then, the vertical split $\mathcal{S}$ wrt. N and $\mathcal{P}$ is $\mathcal{S}' = (\mathcal{D}', loc', Supp)$ with $\mathcal{D}' = \langle \mathcal{N}', \mathcal{L}' \rangle$ where*

$$\mathcal{N}' := \{N_1, N_2\} \uplus (\mathcal{N} \setminus N)$$

$$\mathcal{L}' := \{\, M \xRightarrow{\bar{\sigma}} M' \in \mathcal{L} | M \neq N \wedge M' \neq N \,\} \cup \{\, N_1 \xRightarrow{id} N_2 \,\}$$

$$\cup \{\, M \xRightarrow{\bar{\sigma}} N_1 \mid M \xRightarrow{\bar{\sigma}} N \in \mathcal{L} \,\} \cup \{\, N_2 \xRightarrow{\bar{\sigma}} M \mid N \xRightarrow{\bar{\sigma}} M \in \mathcal{L} \,\}$$

$$loc'(e) = \begin{cases} N_2 & \text{if } loc(e) = N \text{ and } e \in Dom_{\mathcal{D}'}(N_2) \\ N_1 & \text{if } loc(e) = N \text{ and } e \notin Dom_{\mathcal{D}'}(N_2) \\ loc(e) & \text{otherwise} \end{cases}$$

such that $Sig_{\mathcal{D}'}(N_i), i = 1, 2$, are valid signatures and $ax_i, lem_i \subseteq \mathbf{Sen}(Sig_{\mathcal{D}'}(N_i))$, $i = 1, 2$. Conversely, $\mathcal{S}$ is a vertical merge of N_1 and N_2 in $\mathcal{S}'$.

Horizontal Split. Similar to a vertical split we introduce a horizontal split which divides a node into two independent nodes. In order to ensure a valid new development graph, each of the new nodes imports the same theories as the old node and contributes to the same theories as the old node did.

Definition 6 (Horizontal Split). *Let $\mathcal{S} = (\langle \mathcal{N}, \mathcal{L} \rangle, loc, Supp)$ be a* structuring *of (Σ, Ax, Lem), $\mathcal{P} = \{N_1, \ldots, N_k\}$ be a partitioning for some node $N \in \mathcal{N}$ such that each $N_i \in \mathcal{P}$ is lemma independent and $loc^{-1}(N) = dom^N$. The* horizontal split *of $\mathcal{S}$ wrt. N and $\mathcal{P}$ is $\mathcal{S}' = (\mathcal{D}', loc', Supp)$ with $\mathcal{D}' = \langle \mathcal{N}', \mathcal{L}' \rangle$ where*

1. $\mathcal{N}' := \{N_1, \ldots, N_k\} \uplus (\mathcal{N} \setminus N)$

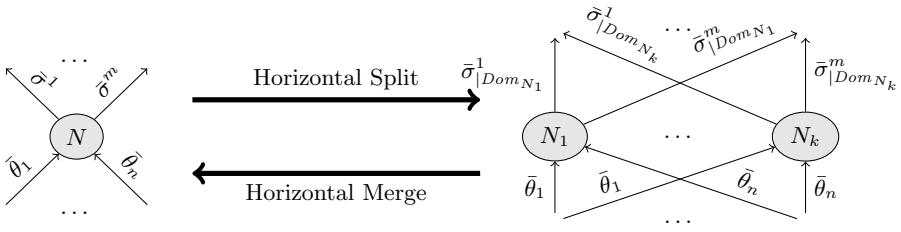

Fig. 2. Horizontal Split and Merge

2. $\mathcal{L}' := \{ M \xRightarrow{\bar{\sigma}} M' \in \mathcal{L} | M \neq N \wedge M' \neq N \}$
$\cup \{ M \xRightarrow{\bar{\theta}} N_i | M \xRightarrow{\bar{\theta}} N \in \mathcal{L}, i \in \{1, \ldots, k\} \}$
$\cup \{ N_i \xRightarrow{\bar{\tau}_{|Dom_{N_i}}} M | N \xRightarrow{\bar{\tau}} M \in \mathcal{L}, i \in \{1, \ldots, k\} \}$
3. $loc'(e) := N_i$ *if* $e \in dom^{N_i}$ *for some* $i \in \{1, \ldots, k\}$ *and* $loc'(e) := loc(e)$ *otherwise.*

such that $Sig_{\mathcal{D}'}(N_i)$ *are valid signatures and* $ax_i, lem_i \subseteq \mathbf{Sen}(Sig_{\mathcal{D}'}(N_i))$ *for* $i = 1, \ldots, k$.

Using the transformation rules, the flat initial theory of our running example can be refactored (cf. Fig. 3). We apply the vertical split rule twice to extract R and the distributivity law, followed by the horizontal-split rule to separate the two instances of a monoid. Finally, we apply the vertical-split rule to extract the extra axioms from $Monoid(R, \times)$. We are left with two copies of a monoid which we like to generalize to a common abstract monoid. This can be achieved by the following rule.

Factorization The factorization rule allows one to merge equivalent concepts into a single generalized concept and then to represent the individual ones as instantiations of the generalized concept. A precondition of this rule is that all individual concepts inherit the same (underlying) theories.

Definition 7 (Factorization). *Let* $\mathcal{S} = (\langle \mathcal{N}, \mathcal{L} \rangle, loc, Supp)$ *be a* structuring *of* (Σ, Ax, Lem). *Let* $K_1, \ldots, K_n, M_1, \ldots, M_p \in \mathcal{N}$ *with* $p > 1$ *such that* $sig^{M_j} \cup ax^{M_j} \neq \emptyset$ *and* $\exists \bar{\sigma}_{i,j}.\ K_i \xRightarrow{\bar{\sigma}_{i,j}} M_j \in \mathcal{L}$ *for* $i = 1, \ldots, n, j = 1, \ldots, p$.

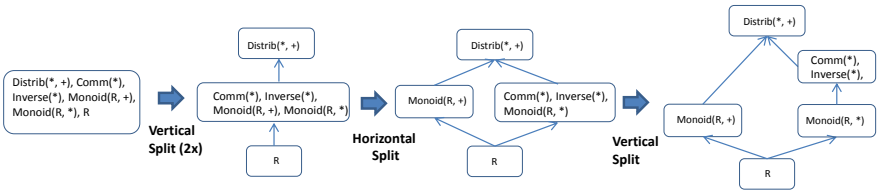

Fig. 3. Applying the split rules to our ring example

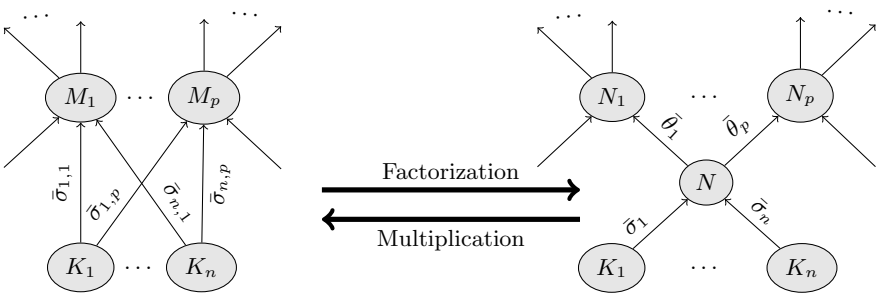

Fig. 4. Factorization and Multiplication (with $\bar{\sigma}_{i,j} := \bar{\theta}_j \circ \bar{\sigma}_i$)

Suppose there are sets sig, ax and lem with $(sig \cup ax \cup lem) \cap Dom_{\mathcal{D}} = \emptyset$ *and pre-signature morphisms* $\theta_1, \ldots, \theta_p$ *and* $\sigma_1, \ldots, \sigma_n$ *such that*
- $\forall e \in Dom_{\mathcal{D}}(K_i).\ \theta_j(\sigma_i(e)) = \sigma_{i,j}(e)$ *and* $\sigma_{i,j}(e) = e \vee \sigma_{i,j}(e) \notin Dom_{\mathcal{D}}$
- $sig^{M_j} \subseteq \theta_j(sig) \subseteq Dom_{\mathcal{D}}(M_j)$, $ax^{M_j} \subseteq \theta_j(ax) \subseteq Dom_{\mathcal{D}}(M_j)$
- $\forall e \in lem$ *holds* $\exists l \in \{1, \ldots p\}.\ \theta_l(e) \in lem^{M_l}$, $\theta_i(e) = \theta_j(e)$ *implies* $i = j$ *and* $\theta_j(e) \in Dom_{\mathcal{D}}$ *implies* $loc(\theta_j(e)) \in M_j$
- *there is a support mapping* $Supp_N$ *for* $ax \cup \bigcup_{i=1,\ldots,n} \sigma_i(Dom_{\mathcal{D}}(K_i))$ *and lem.*

Then $\mathcal{S}' = (\langle \mathcal{N}', \mathcal{L}' \rangle, loc', Supp')$ *is a factorization of* $\mathcal{S}$ *with respect to* $M_1, \ldots, M_p$ *and* $Supp_N$ *iff*

$$\begin{aligned}
\mathcal{N}' :=& \{N\} \cup \{N_j | j \in \{1, \ldots p\}\} \cup \mathcal{N} \setminus \{M_1, \ldots M_p\} \\
& \text{with } N = \langle sig, ax, lem \rangle, N_j = \langle \emptyset, \emptyset, lem^{M_j} \setminus \theta_j(lem) \rangle \\
\mathcal{L}' :=& \{ K \xRightarrow{\bar{\sigma}} K' \in \mathcal{L} | K, K' \notin \{M_1, \ldots M_p\} \\
& \cup \{ K_i \xRightarrow{\bar{\sigma}_i} N | K_i \xRightarrow{\bar{\sigma}_{i,j}} M_j, j \in \{1, \ldots p\}, i \in \{1, \ldots n\}\} \\
& \cup \{ N \xRightarrow{\theta_j} N_j | j \in \{1, \ldots p\}\} \\
& \cup \{ K \xRightarrow{\bar{\tau}} N_j | K \xRightarrow{\bar{\tau}} M_j \wedge (\forall i \in \{1, \ldots n\}. K \neq K_i \wedge \bar{\tau} \neq \bar{\sigma}_{i,j}) \\
& \cup \{ N_j \xRightarrow{\bar{\tau}} K | M_j \xRightarrow{\bar{\tau}} K \in \mathcal{L}, j \in \{1, \ldots p\}\} \\
loc'(x) :=& \begin{cases} N & \text{if } x \in Dom_{\mathcal{D}'}(N) \setminus \bigcup_{i=1,\ldots,n} Dom_{\mathcal{D}'}(K_i) \\ N_j & \text{if } x \in Dom_{\mathcal{D}}(N_j) \text{ and } \forall K \xRightarrow{\bar{\sigma}} N_j.\, x \notin Dom_{\mathcal{D}'}(K) \\ loc(x) & \text{otherwise.} \end{cases} \\
Supp' :=& Supp \cup Supp_N.
\end{aligned}$$

Applying the factorization rule allows us to introduce a generic concept of monoids which is instantiated to obtain both previous copies of a monoid.

The factorization rule only covers a sufficient criterion demanding that each theory imported by a definition link to one concept is also imported via definition links by all other concepts. The more complex case in which a theory is imported

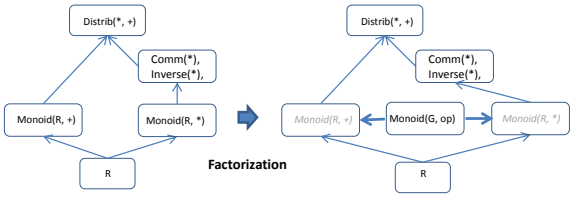

Fig. 5. Factorization of our ring example

via a path of links can be handled by allowing one to shortcut a path in a single global link. This results in the following rule.

Definition 8 (Transitive Enrichment). *Let* $\mathcal{S} := (\langle \mathcal{N}, \mathcal{L} \rangle, loc, Supp)$ *be a structuring of* (Σ, Ax, Lem)*,* $K, N \in \mathcal{N}$ *and there is a path* $K \overset{\bar{\sigma}}{\Longrightarrow} N$ *between both. Then,* $(\langle \mathcal{N}, \mathcal{L} \cup \{ K \overset{\bar{\sigma}}{\Longrightarrow} N \} \rangle, loc, Supp)$ *is a transitive enrichment of* $\mathcal{S}$*.*

Definition links in a development graph can be redundant, if there are alternatives paths which have the same morphisms or if they are not used in any reachable node of the target. We formalize these notions as follows:

Definition 9 (Removable Link). *Let* $\mathcal{S} = (\mathcal{D}, loc, Supp)$ *(*$\mathcal{D} = \langle \mathcal{N}, \mathcal{L} \rangle$*) be a structuring of* (Σ, Ax, Lem)*. Let* $l \in \mathcal{L}$ *and* $\mathcal{D}' = \langle \mathcal{N}, \mathcal{L} \setminus \{l\} \rangle$*.* l *is removable from* $\mathcal{S}$ *and* $\mathcal{S}' = (\mathcal{D}', loc, Supp)$ *is a* reduction *of* $\mathcal{S}$ *iff*

1. $\forall l' : M \overset{\bar{\sigma}}{\Longrightarrow} N$*. if* l' *provides exclusively* $\sigma(e)$ *from some* $e Dom_{\mathcal{D}}(M)$ *then* $e \in Dom_{\mathcal{D}'}(N)$ *and* $l \neq l'$*;*
2. $\forall e \in Dom_{\mathcal{D}}. \forall M \in DGRoots\mathcal{D}$*. if* $loc(e) \overset{\bar{\sigma}}{\Longrightarrow} M$ *then there exists* $M' \in \lceil \mathcal{D}' \rceil$ *such that* $loc(e) \overset{\bar{\sigma}}{\Longrightarrow} M'$*;*
3. $\forall \phi \in Lem_{\mathcal{D}}$*.* $Supp(\phi) \subseteq Dom_{\mathcal{D}'}(N)$ *and* $\forall Sig^{loc}_{\mathcal{D}}(N) \subseteq Dom_{\mathcal{D}'}(N)$*.*

Theorem 1 (Structuring Preservation). *Let* $\mathcal{S} := (\mathcal{D}, loc, Supp)$ *(*$\mathcal{D} = \langle \mathcal{N}, \mathcal{L} \rangle$*) be a structuring of* (Σ, Ax, Lem)*. Then*

1. *every horizontal split of* $\mathcal{S}$ *wrt. some* $N \in \mathcal{N}$ *and partitioning* $\mathcal{P}$ *of* N*,*
2. *every horizontal merge of* $\mathcal{S}$ *wrt. nodes* $\{N_1, \ldots, N_k\} \subseteq \mathcal{N}$*,*
3. *every vertical split of* $\mathcal{S}$ *wrt. some* $N \in \mathcal{N}$ *and partitioning* $\mathcal{P}$ *of* N*,*
4. *every factorization of* $\mathcal{S}$ *wrt. nodes* $M_1, \ldots M_p \in \mathcal{N}$*,*
5. *every multiplication of* $\mathcal{S}$ *wrt.* N*,*
6. *every transitive enrichment of* $\mathcal{S}$*, and*
7. *every reduction of* $\mathcal{S}$

is a structuring of (Σ, Ax, Lem)*.*

5 Refactoring Process

The refactoring rules presented above provide the necessary instruments to externalize the structure inherent in a given flat theory. Nevertheless we need

Char
Sig. $\mathsf{char}, A, B, \ldots, Z$
Ax. $\forall x : \mathsf{char}.\ x = A \vee \cdots \vee Z$
Lem. —

String
Sig. $\mathsf{str}, \mathsf{strnil}, \mathsf{addc}$
Ax. $\forall c : \mathsf{char}, x : \mathsf{str}.\ \mathsf{strnil} \neq \mathsf{addc}(c, x)$
$\forall x : \mathsf{str}.\ x = \mathsf{strnil} \vee \exists c : \mathsf{char}, y : \mathsf{str}.$
$x = \mathsf{addc}(c, y)$
$\forall c, c' : \mathsf{char}, x, x' : \mathsf{str}.\ \mathsf{addc}(c, x) = \mathsf{addc}(c', x')$
$\rightarrow c = c' \wedge x = x', \ldots\}$
Lem. —

Stringops
Sig. $\mathsf{strapp}, \mathsf{strlen}$
Ax. $\mathsf{strlen}(\mathsf{strnil}) = 0$
$\forall c : \mathsf{char}, x : \mathsf{str}.\ \mathsf{strlen}(\mathsf{addc}(c, x)) = \mathsf{succ}(x)$
$\forall x : \mathsf{str}.\ \mathsf{strapp}(\mathsf{strnil}, x) = x, \ldots$
Lem. $\forall x, y, z : \mathsf{str}.\ \mathsf{strapp}(\mathsf{strapp}(x, y), z)$
$= \mathsf{strapp}(x, \mathsf{strapp}(y, z)), \ldots$

Nat
Sig. $\mathsf{nat}, 0, \mathsf{succ}$
Ax. $\forall x : \mathsf{nat}.\ 0 \neq \mathsf{succ}(x)$
$\forall x : \mathsf{nat}.\ x = 0 \vee \exists y : \mathsf{nat}.\ x = \mathsf{succ}(y), \ldots$
Lem. —

Natlist
Sig. $\mathsf{nlist}, \mathsf{natnil}, \mathsf{addn}$
Ax. $\forall n : \mathsf{nat}, x : \mathsf{nlist}.\ \mathsf{natnil} \neq \mathsf{addn}(n, x)$
$\forall x : \mathsf{nlist}.\ x = \mathsf{natnil}$
$\vee\ \exists n : \mathsf{nat}, y : \mathsf{nlist}.\ x = \mathsf{addn}(n, y)$
$\forall n, n' : \mathsf{nat}, x, x' : \mathsf{nlist}.\ \mathsf{addn}(n, x) = \mathsf{addn}(n', x')$
$\rightarrow n = n' \wedge x = x', \ldots$
Lem. —

Natlistops
Sig. $\mathsf{natapp}, \mathsf{nlen}$
Ax. $\mathsf{nlen}(\mathsf{natnil}) = 0$
$\forall n : \mathsf{nat}, x : \mathsf{nlist}.\ \mathsf{nlen}(\mathsf{addn}(n, x)) = \mathsf{succ}(x)$
$\forall x : \mathsf{nlist}.\ \mathsf{natapp}(\mathsf{natnil}, x) = x, \ldots$
Lem. $\forall x, y, z : \mathsf{nlist}.\ \mathsf{natapp}(\mathsf{natapp}(x, y), z)$
$= \mathsf{natapp}(x, \mathsf{natapp}(y, z)) \ldots$

Fig. 6. Heuristic motivated partitioning of a flat theory

appropriate heuristics to determine suitable partitions for horizontal or vertical splits, in order to group together strongly related entities and afterwards make explicit analogous groupings of entities by factorization. One heuristic is, for instance, to group together all axioms and lemmas exclusively devoted to a specific sort. E.g. all axioms about characters, or all axioms about natural numbers. From there we carve out those entities about a different sort *including* an already classified sorts, and so on. Examples for these are strings or natural numbers, and in the next iteration lists of strings (cf. Fig. 6). These identified subsets can then be further partitioned into those defining the basic datatype and the (inductive) functions and predicates over these. This separates the definition of lists of natural numbers from the append functions on these, for instance.

Such heuristics guide the application of the horizontal and vertical split rules. From there we can identify nodes with isomorphic local entries and, using the transitivity and reduction rules together with further applications of the split rules try to enable the application of the factorization rule.

We illustrate the refactoring process with the help of a further example, where we start with flattened theories, and step by step carve out the intrinsic structure. The example from Fig. 6 is about lists of natural numbers, strings formed over characters and length functions operating on such lists. All the signature symbols, axioms, and lemmas occur in a single node in the initial development graph. For sake of readability we partition the set of these entities into pairwise disjunct subsets and name them accordingly (cf. [8])

Fig. 7 illustrates the structure formation process for this example. In the first step the definitions of Nat and Char are separated from the rest by applying the vertical split rule twice. In the next steps we form the individual theories of String and Stringops respectively. Notice that Stringops includes strlen counting characters in a string such that Stringops has to import Nat. After applying again a vertical split we obtain the theories of Natlist and Natlistops. Since the local axioms of String and Natlist are renamings of each other we factorize these local axioms to a new theory of generic lists.

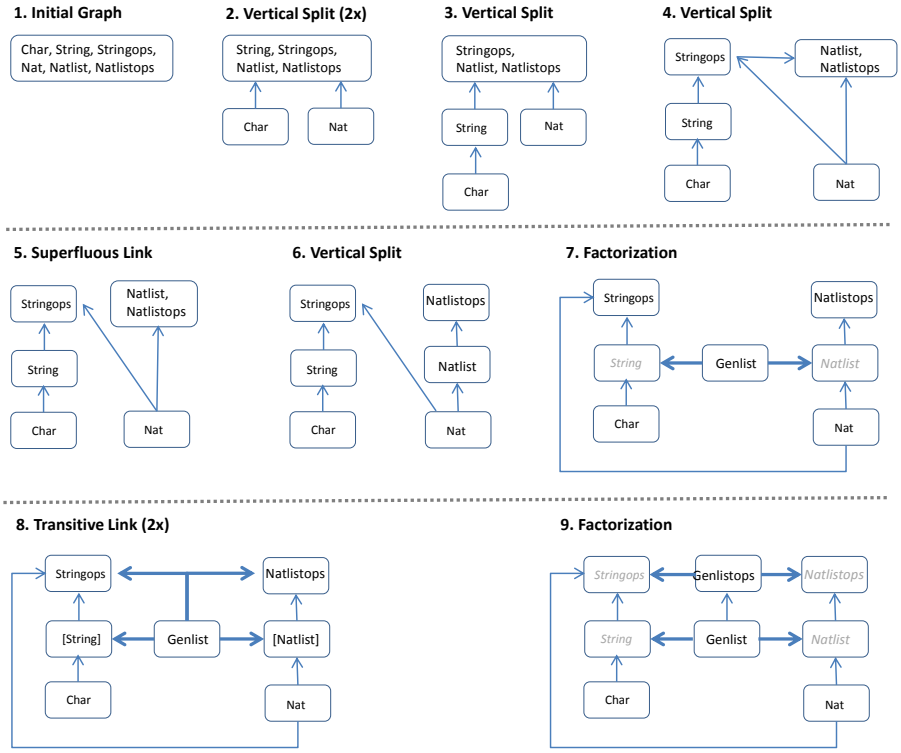

Fig. 7. Lists

This node consists of the renamed local axioms of String (or Natlist, respectively) plus the necessary signature definitions to obtain a well-formed development graph node. This theory is imported via definition links to String and Natlist using corresponding pre-signature morphisms to map it to list of chars and numbers, respectively. Both, String and Natlist have now empty local signatures and axioms. In Fig 7 we indicate those nodes with empty local signature, empty local axioms and empty local lemmata by a light-gray color of the node name.

Genlist
Sig. list, nil, add
Ax. $\forall n : \mathsf{elem}, x : \mathsf{list}.\ \mathsf{nil} \neq \mathsf{add}(n, x)$
$\forall x : \mathsf{list}.\ x = \mathsf{nil}$
$\vee\ \exists n : \mathsf{elem}, y : \mathsf{list}.\ x = \mathsf{add}(n, y)$
$\forall n, n' : \mathsf{elem}, x, x' : \mathsf{list}.\ \mathsf{add}(n, x) = \mathsf{add}(n', x')$
$\rightarrow n = n' \wedge x = x', \ldots$
Lem. —

6 Related Work

There is related work to (re-)structure ontologies (after flattening) following locality criteria into modules containing all knowledge about specific concepts. However, since ontology languages have simple imports without renamings, duplicated sub-ontologies that are for instance equal up to renaming remain hidden.

To excavate these ontologies requires other heuristics than pure logic based ones to guide the structuring process, and therefore we deliberately did not impose specific criteria how to apply the rules beyond the minimal conditions that the dependency relations among concepts, relations and facts and derived facts and relations is preserved. It is easy to see that the modularizations of ontologies not including derived lemmas obtained by using the locality criteria from [9] can be constructed with our rules by starting from a single flattened ontology and singling out the modules.

7 Conclusion

Based on the definition of structurings as concise development graphs, we presented transformation rules that allow one to make explicit common structures hidden in a flat theory in terms of development graphs. It provides a framework to modularize flattened ontologies in a useful way as illustrated by two simple examples. An implementation is planned to provide an interactive tool to modularize and factorize large ontologies as well as (semi-)automatic procedures.

References

1. S. Autexier and D. Hutter. Mind the gap - maintaining formal developments in MAYA. In *Festschrift in Honor of J.H. Siekmann*. Springer, LNCS 2605, 2005.
2. S. Autexier, D. Hutter, and T. Mossakowski. Change management for heterogeneous development graphs. In S. Siegler and N. Wasser, editors, *Verification, Induction, Termination Analysis (Festschrift for Christoph Walther)*, volume 6463 of *LNAI*, pages 54–80. Springer, 2010.
3. J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39:95–146, 1992. Predecessor in: LNCS 164, 221–256, 1984.
4. D. Hutter. Management of change in verification systems. In *Proceedings 15th IEEE International Conference on Automated Software Engineering, ASE-2000*, pages 23–34. IEEE Computer Society, 2000.
5. O. Kutz and T. Mossakowski. A modular consistency proof for DOLCE. In *The Association for the Advancement of Artificial Intelligence (AAAI-2011)*, 2011.
6. T. Mossakowski, S. Autexier, and D. Hutter. Development graphs - proof management for structured specifications. *Journal of Logic and Algebraic Programming, special issue on Algebraic Specification and Development Techniques*, 67(1-2):114–145, april 2006.
7. I. Niles and A. Pease. Towards a standard upper ontology. In *Formal Ontology in Information Systems (FOIS-2001)*. ACM Press, 2001.
8. I. Normann and M. Kohlhase. Extended formula normalization for ϵ -retrieval and sharing of mathematical knowledge. In *Towards Mechanized Mathematical Assistants (Calculemus/MKM)*. Springer, LNCS 4573, 2007.
9. C. D. Vescovo, B. Parsia, U. Sattler, and T. Schneider. The modular structure of an ontology: Atomic decomposition. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'10)*, pages 2232–2237, 2010.
10. A. Voronkov. Inconsistencies in ontologies. In *JELIA 2006*. Springer LNCS 4160, 2006.

Characterizing Modular Ontologies

Sarra Ben Abbès[1], Andreas Scheuermann[2], Thomas Meilender[3], and Mathieu d'Aquin[4]

[1]Laboratoire d'Informatique de Paris Nord (UMR 7030), CNRS, Paris 13 University, Sorbonne Paris Cité, France. E-mail: sarra.benabbes@lipn.univ-paris13.fr
[2]University of Hohenheim, Information Systems 2, 70599 Stuttgart, Germany. Email: andreas.scheuermann@uni-hohenheim.de
[3]UHP-Nancy 1, LORIA (UMR 7503 CNRS-INPL-INRIA-Nancy 2-UHP), France. Email: thomas.meilender@loria.fr
[4]Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK. E-mail: m.daquin@open.ac.uk

Abstract. Since large monolithic ontologies are difficult to handle and reuse, ontology modularization has attracted increasing attention. Several approaches and tools have been developed to support ontology modularization. Despite these efforts, a lack of knowledge about characteristics of modularly organized ontologies prevents further development. This work aims at characterizing modular ontologies. Therefore, we analyze existing modular ontologies by applying selected metrics from software engineering in order to identify recurring structures, i.e. patterns in modularly organized ontologies. The contribution is a set of four patterns which characterize modularly organized ontologies.
Keywords: modularization, patterns, modules, ontology.

1 Introduction

Difficulties in reusing and maintaining large monolithic ontologies have resulted in an increasing interest in modularizing ontologies. In the past, several approaches and tools (e.g., SWOOP[1], NeOn Toolkit[2]) have been proposed to support the modularization of ontologies. Each of these approaches and tools respectively incorporates its own definition and notion of modular ontologies and criteria underpinning ontology modularization [5]. This proliferation is mainly due to the fact that the area of ontology modularization appears to be still in its infancy. Thus, it lacks the mature, well-defined, well-understood, and commonly agreed upon definitions and concepts as sociated with modularization in the area of software engineering [19]. This lack of knowledge about ontology modularization and, particularly, the lack of knowledge about characteristics of modular ontologies prevents its further development. Being able to characterize modular

[1] `http://www.mindswap.org/2004/`
[2] `http://neon-toolkit.org/wiki/Main_Page`

ontologies would allow for (1) comparing different modularization approaches, (2) assessing the quality of modular ontologies with respect to manually modularized ontologies, (3) customizing ontology modularization by providing modularization criteria, and (4) improving the area of ontology modularization as a whole.
The goal of this work is to characterize modularly organized ontologies to contribute to a better understanding of ontology modularization. For this purpose, we (1) extract a number of existing ontologies from the web, (2) select and adopt metrics from software engineering in order to both (3) identify recurring structures (patterns) in modularly organized ontologies and (4) characterize the identified patterns. The contribution is a set of four patterns, which characterize modularly organized ontologies.
The rest of this paper is organized as follows: Section 2 provides an introduction to ontology modularization and reviews related work. Section 3 reports on the research design whereas section 4 presents and discusses the results. Section 5 draws a conclusion and points to future avenues of research.

2 Ontology Modularization

2.1 Modular Ontologies

The main idea of modular ontologies originates from the general notion of modular software in the area of software engineering. Correspondingly, ontology modularization can be interpreted as decomposing potentially large and monolithic ontologies into (a set of) smaller and interlinked components (modules). Therefore, an ontology module can be considered as a loosely coupled and self-contained component of an ontology maintaining relationships to other ontology modules. Thereby, ontology modules are themselves ontologies [4].
In general, ontology modularization aims at providing users of ontologies with the knowledge they require, reducing the scope as much as possible to what is strictly necessary. In particular, ontology modules (1) facilitate knowledge reuse across various applications, (2) are easier to build, maintain, and replace, (3) enable distributed engineering of ontology modules over different locations and different areas of expertise, and (4) enable effective management and browsing of modules [12].

2.2 Approaches for Ontology Modularization

In recent years, the problem of ontology modularization has attracted more and more attention and, thus, several different approaches for modularizing ontologies appeared. These approaches can be classified in two main categories.
The first main category comprises approaches that focus on the composition of existing ontologies by means of integrating and mapping ontologies. On the one hand, approaches addressing integration of existing ontologies are owl:import, partial semantic import, e.g., [8, 5], package-based description logics, e.g., [2].

On the other hand, mapping approaches basically aim at (inter-)linking sets of ontology modules. The following approaches can be assigned to these two formalisms: distributed description logics, e.g., [17, 3], ε-connections, e.g., [15, 10]. Other approaches establish the relationship between various modular ontology formalisms [9] in order to have special syntax in the ontology languages for a modeling perspective.
The second main category comprises approaches for modularizing ontologies in terms of ontology partitioning and ontology module extraction. On the one hand, ontology partitioning aims at splitting up an existing ontology into a set of ontology modules. Approaches for partitioning ontologies are proposed by [13, 11] whereas [18] proposes a tool. On the other hand, ontology module extraction, which is also called segmentation [16] or traversal view extraction [14], aims at reducing an ontology to its relevant sub-parts. Approaches for ontology module extraction are the subject of [14, 16], and the PROMPT tool [14]. More details of this category of approaches are discussed in [5].

2.3 Criteria for Ontology Modularization

Criteria for modularizing ontologies generally aim at characterizing modular ontologies. To the best of our knowledge, only [5] explicitly deals with criteria for ontology modularization. Therefore, [5] distinguishes between criteria originating in software engineering, logical criteria, local criteria, structural criteria, quality of modules, and relations between modules. First, criteria from software engineering comprise encapsulation and coherence whereas logical criteria include local correctness and local completeness. Structural criteria, which are also discussed by [6], focus on size and intra-module coherence. It is proposed to determine the quality of modules in terms of module cohesion, richness of the representation, and domain coverage. At least, to assess the relation between modules the criteria of connectedness, redundancy, and inter-module distance can be applied. Against this background, the evaluation of ontology modularization respectively applies a subset of the proposed set of criteria with respect to different scenarios and ontology modularization techniques.
Based on best practices in Ontology Engineering, ontology design patterns (ODPs) simplify ontology design by providing a "modelling solution to solve recurrent ontology design problems" [1]. Several types of ODPs has already been identified, e.g., logical patterns that are used to solve problems of expressivity, or naming patterns that are conventions for naming elements. Among these types, architectural ontology design patterns (AODPs) aim at describing the overall shape of the ontology. More precisely, external AODPs describe the modular architecture of an ontology by considering a modular ontology as an ontology network. Involved ontologies are considered as modules and are connected by the import operation. A semantic web portal[3] has been proposed as a repository for ODPs. Unfortunately, to our knowledge, no work has been done on proposing and describing external AODPs.

[3] `http://ontologydesignpatterns.org`

3 Approach

In order to characterize reoccurring structures in modularly organized ontologies, the following approach establishes a methodological basis to guide the research program of this work. This approach comprises six subsequent steps:

1. **Search step**: the goal of the first step is to gather modularly organized ontologies. We use the Semantic Web gateway Watson[4] to search for available modular ontologies from the WWW. The search query focused on import-relationships between ontologies covering the same domain. The result is a set of 77 modularly organized ontologies.
2. **Cleaning up step**: the second step aims at cleaning up the initial search results in order to establish a thorough basis for further experiments. This is necessary because the set of 77 modular ontologies is afflicted with redundancies and incompleteness. This results in a set of 38 modular organized ontologies constituting a thorough basis for characterizing ontology modularization.
3. **Selection of metrics step**: the third step selects a set of appropriate metrics to characterize modularly organized ontologies. The modular ontologies could be described by various indicators such as the distribution of classes, the network of links between modules, the number of internal links in modules, etc. In general, the literature provides a plethora of various metrics, which could be applied for characterising modular ontologies. As a starting point, this work focuses on metrics originating in the area of software engineering, due to its maturity. In particular, this work adopts the following metrics from software engineering, which are easier to compute, in order to characterize modular ontologies [7]: (i) **size of the module**: the number of classes and properties (object and datatype properties), (ii) **cohesion of the module**: this metric is a value which is between 0 and 1 and is specified as follows:
   * Hierarchical Class Cohesion (HCC): the number of direct and indirect hierarchical class links.

     $$HCC = \frac{2*(NdHC+NidHC)}{NC^2-NC}$$

     where: $NdHC$: Number of direct Hierarchical relationships between Classes, $NidHC$: Number of indirect Hierarchical relationships between Classes, and NC: Number of Classes.
   * Role Cohesion (RC): the number of direct and indirect hierarchical role links.

     $$RC = \frac{2*(NdR+NidR)}{NRoles^2-NRoles}$$

     where: NdR: Number of direct roles between Classes, $NidR$: Number of indirect roles between Classes, and $NRoles$: Number of Roles.

[4] `http://watson.open.ac.uk/`

* Object Property Cohesion (OPC): the number of classes which have been associated through the particular object property (domain and range).

$$OPC = \frac{2*\sum_{i=1}^{NRoles} NdC(r_i)*NrC(r_i)}{NRoles*(Nc^2-NC)}$$

where: $NdC(r_i)$: Number of ontology Classes in the domain of the role r_i, $NrC(r_i)$: Number of ontology Classes in the range of the role r_i, NC: Number of Classes, and $NRoles$: Number of Roles.

The cohesion measure is computed as follows:

$$Cohesion = \frac{\alpha*HCC+\beta*RC+\delta*OPC}{\alpha+\beta+\delta}$$

where: α, β and δ specify the impact of each type of hierarchical class, role or object property cohesion. In our case, we choose $\alpha = \beta = \delta = 1$.

(iii) **coupling of the module**: it takes an estimation of the inter-dependency of different modules and is specified as follows:

* Hierarchical class dependency (HCD): the number of all direct and indirect hierarchical class relationships to foreign ontologies.

$$HCD = \frac{1}{2} * (\frac{NedHC}{NdHC} + \frac{NeidHC}{NidHC})$$

where: $NedHC$: Number of direct Hierarchical class dependencies between local classes and external classes, and $NeidHC$: Number of indirect Hierarchical class dependencies between local classes and external classes.

* Hierarchical role dependency (HRD): the number of all direct and indirect hierarchical role relationships to foreign ontologies.

$$HRD = \frac{1}{2} * (\frac{NdHR}{NdHR} + \frac{NeidHR}{NidHR})$$

where $NdHR$: Number of direct roles dependencies between local classes and external classes, and $NeidHR$: Number of indirect roles dependencies between local classes and external classes.

* Object property dependency (OPD): the number of roles that associate external classes to local ones.

$$OPD = \frac{NeRoles}{NRoles}$$

where: $NeRoles$: Number of all roles that have an external class in their domain or range, $NRoles$: Number of all existing roles in the ontology.

* Axiom dependency (AD) : a role or a class is associated to an external ontological element through an inclusion axiom.

$$AD = \frac{\sum_{i=1}^{NAxioms} externalAssociationNumber(axm_i)}{\sum_{i=1}^{NAxioms} LS(axm_i*RS(axm_i)}$$

where: $LS(axm)$: the size of the left sides of the axiom *axm*, $RS(axm)$: the size of the right sides of the axiom *axm*, $LSE(axm)$: the number of external elements in the left sides of the axiom *axm*, $RSE(axm)$: the number of external elements in the right sides of the axiom *axm*, and $externalAssociationNumber(axm)$: the number of all external ontological elements that have been associated through the axiom *axm* to internal elements. $externalAssociationNumber(axm) = LSE(axm_i)*RS(axm_i) + LS(axm_i) * RSE(axm_i)$ - $LSE(axm_i) * RSE(axm_i)$.

The coupling measure is computed as follows:

$$Coupling = \frac{\alpha * HCD + \beta * HRD + \delta * OPC + \gamma * AD}{\alpha + \beta + \delta + \gamma}$$

where, in our case, $\alpha = \beta = \delta = \gamma = 1$

4. **Metrics implementation step**: the fourth step implements the selected metrics. The computation was performed by the OWL API[5] and the reasoner HermiT[6]. This step sets up the (technical) evaluation framework.
5. **Analysis step**: the fifth step is the analysis of the basic population of modularly organized ontologies.
6. **Result step**: the sixth step involves synthesis and discussion of the results from the analysis in order to characterize modular ontologies.

4 Results and Discussion

A set of four patterns, which characterize *M*odular *O*ntologies *MO*, are proposed using previous metrics (size, cohesion and coupling). This section presents and discusses the results and the characteristics of each kind of pattern.

4.1 Pattern type 1: 1 module importing n modules

Pattern 1 contains one module which imports n other modules. For instance (Figure 1), the module *WildNET.owl* imports several modules such as *Animal.owl*, *AnimalSighting.owl*, *BirdObservers.owl*, *Birds.owl*, etc. The pattern that we propose conforms to an **aggregation**. This pattern establishes a relationship between a single module and a set of modules in the same ontology. This link is unidirectional. Applying the size metric (Table 1), the first part of the ontol-

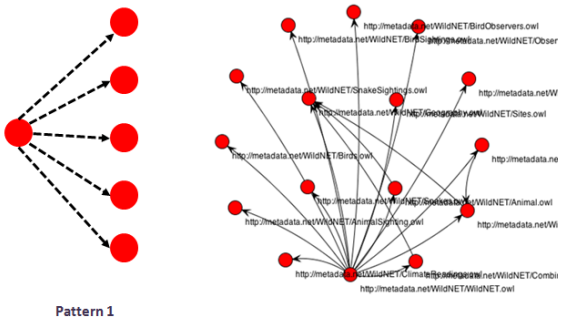

Fig. 1. 1 module importing n modules

ogy (one module) is very small (the module *WildNET.owl* contains 0 concepts)

[5] `http://owlapi.sourceforge.net/`
[6] `http://hermit-reasoner.com/`

and the second part (N modules) is not structured and is respectively bigger in size. Applying the cohesion and coupling metrics (see Table 1), Pattern 1 has a high cohesion compared to the coupling metric. We consider the pattern cohesion metric to be an indicator of the degree to which the elements in the module belong together. The idea of this pattern is that the concepts grouped in an ontology should be conceptually related for a particular domain in order to achieve common goals.

Metrics		Size	Cohesion	Coupling
MO_{11}	WildNET.owl	0	0,25	0
	SnakeSightings.owl	18331	1	0,25
	Snakes.owl	95	0,5	0,25
	Sites.owl	8	1	0,25
	Observers.owl	7	0,53	0,25
	Geography.owl	16	0,41	0,25
	Combined.owl	3497	0,67	0,12
	ClimateSensors.owl	255	0,61	0,6
	ClimateReadings.owl	63	0,5	0,25
	Climate.owl	24	0,61	0,15
	BirdSites.owl	437	1	0,25
	BirdSightings.owl	10401	1	0,5
	Birds.owl	1745	0,5	0,25
	BirdObservers.owl	303	0,5	0,25
	AnimalSighting.owl	26	0,66	0,25
	Animal.owl	9	0,82	0,11

Table 1. Results of Pattern 1 to n

4.2 Pattern type 2: n modules importing 1 module

Pattern 2 contains n modules, which respectively import one module. For instance (Figure 2), there are three independent modules importing one module, which containes general knowledge (*biopax-level1.owl*). The pattern that we propose corresponds to **inheritance**. This pattern establishes a correspondence between a set of modules and a single module in the same ontology. This correspondence is unidirectional. Applying the three metrics (Table 2), the first part of pattern 2 (n modules), *biopax-example-ecocyc-glycolysis.owl*, *biopax-example-Xwnt-b-catenin.owl* and *Xwnt-b-catenin_xref* have a high coupling metric with regard to the second part of the pattern (one module) *biopax-level1.owl*. This means that pattern 2 is characterized by the interconnections between modules. The degree of coupling depends on how complicated the connections are and on the type of connections between modules. As we can see, the second part of pattern 2 has a high cohesion because it encloses all other modules, which are strongly related.

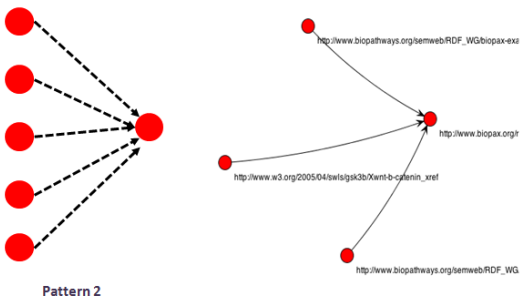

Fig. 2. n modules importing 1 module

Metrics		Size	Cohesion	Coupling
MO_{21}	biopax-example-ecocyc-glycolysis.owl	2139	0,20	0,72
	biopax-example-Xwnt-b-catenin.owl	236	0	0,2
	biopax-level1.owl	285	0,25	0,13
	Xwnt-b-catenin_xref	265	0	0,2

Table 2. Results of Pattern n to 1

4.3 Pattern type 3: n modules importing n-1 modules

Pattern 3 contains n modules, which import n-1 modules. For instance (Figure 3), we have three dependent modules: *dublincore.owl*, *terms.owl* and *dcmitype.owl*. The correspondence between modules is bidirectional. The distinguishing characteristic of Pattern 3 is that the n modules each import each other. Applying size,

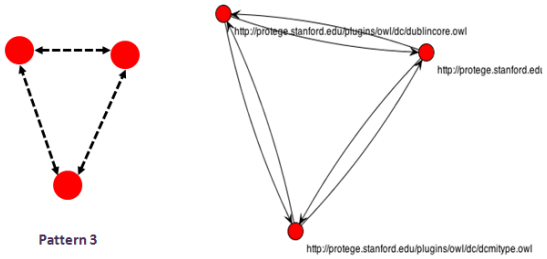

Fig. 3. n modules importing n-1 modules

cohesion and coupling metrics (Table 3), the module *dublincore.owl* has a small size (0 concepts) with regard to other modules *dcmitype.owl* and *terms.owl*. All modules have the same degree of relatedness of concepts (cohesion) 20%. The

coupling metric of the module *dublincore.owl* is null. In this case, pattern 3 is transformed to pattern 1 and has the same characteristics.

Metrics		Size	Cohesion	Coupling
MO_{32}	dcmitype.owl	21	0,20	0,02
	dublincore.owl	0	0,20	0
	terms.owl	47	0,20	0,25

Table 3. Results of Pattern n to n-1

4.4 Pattern type 4: Pattern mix

Pattern 4 combines all previous patterns (Patterns 1, 2 and 3). For instance (Figure 4), we find partterns 1 (5 * Pattern 1) and 2 (3 * Pattern 2). The proposed pattern is **pattern mix**. The correspondence can be undirectional and bidirectional. The major characteristic of this type of pattern is the highest

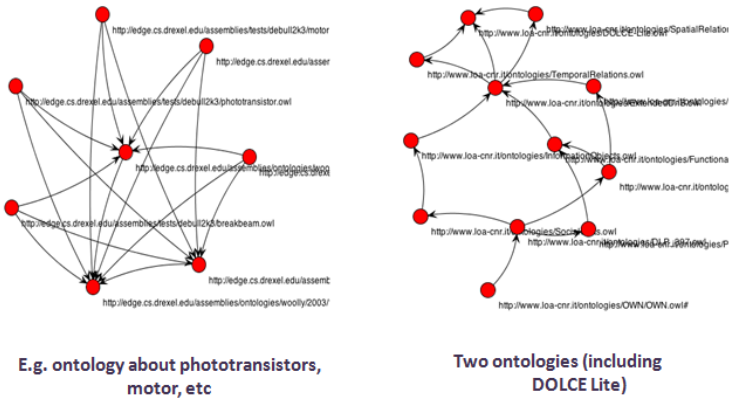

Fig. 4. Combination of patterns 1, 2, and 3

coupling metric with regard to the cohesion one. Two modular ontologies *iso-metadata* and *iso-19115*, have the same size, cohesion and coupling but they do not have a relationship like *import*.

4.5 Occurrence of Pattern Types

Having introduced and defined four types of patterns in order to characterize modularly organized ontologies, we consider how often these types of patterns

Metrics		Size	Cohesion	Coupling
MO_{41}	iso-metadata	2214	0,02	0,15
	iso-19108	159	0,08	0,16
	iso-19103	224	0,15	0,16
	iso-19115	2214	0,02	0,15

Table 4. Results of Pattern mix

respectively occur. Figure 5 provides an overview of the occurrence of the different types of patterns in a population of 38 modularly organized ontologies. It is

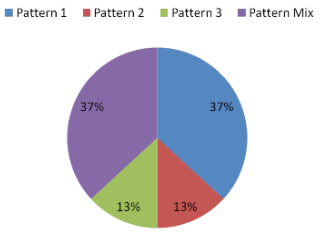

Fig. 5. Occurrence of Pattern Types

interesting to observe that pattern type 1 accounts for about 37%. The reason for this may be the fact that this type of pattern appears to be very intuitive. It could therefore be concluded that it implicitly constitutes the rationale underlying a large part of (semi-)automatic or manual approaches for modularizing ontologies. Similarly, pattern type 4 also accounts for about 37% of the basic population, i.e. 14 modularly organized ontologies. Pattern type 4 combines Pattern 1, Pattern 2, and Pattern 3. On the one hand, it is obvious that not all modularly organized ontologies have a rather straightforward structure, which could be easily characterized. This is especially true when assuming (semi-)automatic or manual modularizing approaches, which do not use clear and precisely defined criteria. And even when these criteria are clearly and precisely defined, the modularly organized ontologies could also have such a structure depending on the overall purpose of modularization. On the other hand, it is interesting to see that even more complex structures can (almost completely) be characterized by more simple and straightforward structural forms. Moreover, pattern type 2 and pattern type 3 equally account for about 13%, i.e. 5 modularly organized ontologies. This is particularly interesting because pattern type 2 is reasonable. This is due to the fact that it appears obvious that there exists an ontology that is of significance for several other ontologies. On the contrary, pattern type 3 is much less reasonable than pattern type 2. It is really hard to understand why ontology modules respectively import each other.

In this context, it can be observed that domain ontologies combine a clear structure and organization. This means that modularization of domain ontologies tend to rely on pattern type 1 or pattern type 2. In contrast, it appears that top-level ontologies (which represent relevant knowledge to a particular domain such as medical domain) have a less straightforward structure and organization particularly when compared to domain ontologies (which represent upper (generic) ontologies, covered the knowledge of many domain types such as Biomedical ontology, Dolce). An example for this is *dublincore.owl*, *terms.owl* and *dcmitype.owl*, which can be characterized by pattern type 3 (Figure 3).

5 Conclusion and Future Work

This work aims at characterizing modularly organized ontologies to contribute to a better understanding of ontology modularization. We introduced the notion of modular ontologies, reported on approaches for ontology modularization, and reviewed existing efforts to characterize modular ontologies. To characterize modular ontologies, we followed an approach comprising six consecutive steps. This approach mainly includes the extraction and selection of modular ontologies, the selection of a set of metrics from software engineering to analyse modular ontologies, and the evaluation of the analysis results. The evaluation results in a set of four patterns, which allow for characterizing the modular organization of ontologies. These patterns show amongst other things that modularly organized domain ontologies have a clear structure whereas top-level ontologies tend to have a rather confusing modular organisation.
In the future work, we aim at using firstly further Semantic Web gateways such as Falcons or Swoogle to identify and extract additional ontologies to gain a larger basic population. Second, extending the set of metrics and applying them to the ontologies should provide further insights to modular ontologies. Third, it would be interesting to create a comparison framework to conduct experiments with different modularization approaches, comparing them to each other or to manually modularized ontologies.

Acknowledgment

The authors would like to thank the organizers of Summer School on ontology engineering and the Semantic Web 2001 (SSSW'2011).

References

1. Gangemi A. and Presutti V. *Ontology Design Patterns*, pages 221–243. Springer, Berlin, 2009.
2. Jie Bao, George Voutsadakis, Giora Slutzki, and Vasant Honavar. Package-based description logics. In *Modular Ontologies*, pages 349–371. Springer, 2009.

3. Alexander Borgida and Luciano Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In *On the Move to Meaningful Internet Systems*, pages 36–53, London, UK, 2002. Springer-Verlag.
4. Mathieu d'Aquin, Peter Haase, Sebastian Rudolph, Jérôme Euzenat, Antoine Zimmermann, Martin Dzbor, Marta Iglesias, Yves Jacques, Caterina Caracciolo, Carlos Buil Aranda, and Jose Manuel Gomez. D1.1.3: Neon formalisms for modularization: Syntax, semantics, algebra. deliverable 1.1.3, NeOn Integrated Project, 2008.
5. Mathieu d'Aquin, Anne Schlicht, Heiner Stuckenschmidt, and Marta Sabou. Modular ontologies. chapter Criteria and Evaluation for Ontology Modularization Techniques, pages 67–89. Springer-Verlag, Berlin, Heidelberg, 2009.
6. Frederico Luiz Gonçalves de Freitas, Zacharias Candeias Jr., and Heiner Stuckenschmidt. Towards checking laws' consistency through ontology design: The case of brazilian vehicles' laws. *JTAER*, 6:112–126, 2011.
7. Faezeh Ensan and Weichang Du. *A Modular Approach to Scalable Ontology Development*, page 79. Springer Science+Business Media, 2010.
8. Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. A logical framework for modularity of ontologies. In *JCAI-2007*, 2007.
9. Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Extracting modules from ontologies: A logic-based approach. In *Modular Ontologies*, pages 159–186. 2009.
10. Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Working with multiple ontologies on the semantic web. In *International Semantic Web Conference*, pages 620–634. Springer, 2004.
11. Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In *20th International Conference on Principles of Knowledge Representation and Reasonin*, pages 198–209. AAAI Press, 2006.
12. Mustafa Jarrar. *Towards Methodological Principles for Ontology Engineering*. PhD thesis, Vrije Universiteit Brussel, 2005.
13. Bill MacCartney, Sheila McIlraith, Eyal Amir, and Tomás E. Uribe. Practical partition-based theorem proving for large knowledge bases. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 89–96, San Francisco, USA, 2003. Morgan Kaufmann Publishers Inc.
14. Natalya F. Noy and Mark A. Musen. Specifying ontology views by traversal. In *International Semantic Web Conference*, volume 3298/2004, pages 713–725. Springer Berlin, 2004.
15. Ana Armas Romero, Bernardo Cuenca Grau, and Ian Horrocks. Modular combination of reasoners for ontology classification. In *Description Logics*, 2012.
16. Julian Seidenberg and Alan Rector. Web ontology segmentation: analysis, classification and use. In *Proceedings of the 15th international conference on World Wide Web*, pages 13–22, New York, USA, 2006. ACM.
17. Luciano Serafini and Andrei Tamilin. Drago: Distributed reasoning architecture for the semantic web. In *ESWC*, pages 361–376. Springer, 2005.
18. Heiner Stuckenschmidt and Michel Klein. Structure-based partitioning of large concept hierarchies. In *International Semantic Web Conference*, pages 289–303, 2004.
19. Kevin J. Sullivan, William G. Griswold, Yuanfang Cai, and Ben Hallen. The structure and value of modularity in software design. In *Proceedings of the 8th European software engineering conference*, pages 99–108, New York, USA, 2001. ACM.

Combining three Ways of Conveying Knowledge: Modularization of Domain, Terminological, and Linguistic Knowledge in Ontologies

Thierry Declerck[1] and Dagmar Gromann[2]

[1] DFKI GmbH, Language Technology Department,
Stuhlsatzenhausweg 3, D-66123 Saarbruecken, Germany
declerck@dfki.de
[2] Vienna University of Economics and Business,
Nordbergstrasse 15, 1090 Vienna, Austria
dgromann@wu.ac.at

Abstract. Recently, an overall trend towards increasing complexity of ontologies could be observed, not only in terms of domain modeling, where the complexity should correspond to the information to be modeled, but also as regards the addition of further information, which could be modeled as external resources to the domain model and linked to its relevant elements. This concerns the addition of terminological and linguistic information to the description of classes and properties of ontologies. To respond to this development, we propose a functional approach to the modularization of ontologies, based on terminological, linguistic, and conceptual functions each module fulfills. Only the conceptual elements and their structural properties should remain in the domain model, whereas the formalized terminology and linguistics are described in independent modules referencing the domain models. We provide examples of such complexity in Knowledge Representation systems, discuss related work, and present our approach to modularization in detail.

Keywords: ontology, terminology, linguistics, lexicon, LabelNet, SKOS, TBX, TMF, lemon

1 Introduction

Nowadays, ontologies in general not only contain domain knowledge but further information central to various tasks of ontology-based systems. For instance, terminological and linguistic details are substantially different in nature from the former and usually encoded in labels adjoined to IDs of classes and properties.

There is a growing realization among many researchers that it might not be the best practice to encapsulate such information within the description of classes and properties of domain ontologies. Proposals have already been made for the separation of terminology and lexicon from domain ontologies and for strategies on the linking of this information to the elements of the domain model

in a more principled way [1–6]. Our approach to modularization can be considered functional, as it is based on the functions the terminological and linguistic elements used in the context of domain models fulfill. Several tasks such as supporting Information Systems (IS), semantic annotation, lexicographic applications, translation, localization among many others benefit from encapsulated and reusable functions as presented herein.

The need to cull content of labels in ontologies has increased with more possibilities to linguistically process labels, adding linguistic annotations to their textual content and thereby more complexity to the ontology. As a result, reusability and sharing of the information accumulated is considerably impeded since navigation through the entire ontology is required in order to find linguistically annotated terms that are relevant to ontology-driven applications.

Therefore, following a series of similar proposals [1–3], extending and specifying some points made, we suggest a strict modularization of domain ontologies in a class hierarchy, a terminology, and a linguistic component, all represented in RDF/OWL and related to each other by means of the Simple Knowledge Organization Scheme of the W3C (SKOS) and similar linking mechanisms. Thus, a lexical entry can be used by several terminologies, terms of which are employed in different specific ontologies.

The proposed model largely facilitates the detection of interrelations among ontologies, rendering the formation of new ontologies on the basis of existing independently built ones faster and less complicated, because the model strips ontologies to their core and most essential elements. It equally aims at more compact terminologies and lexicons used in relation with domain modeling, since variants of these can be more easily detected and collapsed onto harmonized sets. Thus, our three-module system represents a mechanism for increasing flexibility in reusing ontologies as well as domain-specific lexicons and terminologies.

2 Steadily Growing Complexity of Ontologies

A class defined in the RadLex ontology[3] serves to exemplify the growing complexity in ontologies. As can be seen in the example below, the class RID 13218 contains all information about its superordinate class and the related properties. Furthermore, information on natural language expressions associated with the class (*synonym, NonEnglish_Name, Preferred_Name, ORIG_Preferred_Name, Definition*) as well as other knowledge sources, i.e., FMAID 67112, were accumulated to form one single ontology class. The knowledge source refers to the Foundational Model of Anatomy (FMA)[4]. Upon looking at the entry in the FMA ontology, it can quickly be inferred that elements have just been duplicated, such as the definition, synonym, the (German) Non-English part and the label (preferred name).

[3] Version 3, http://bioportal.bioontology.org/ontologies/2027?p=terms

[4] The URL for the indicated ID is `http://bioportal.bioontology.org/ontologies/44507/?p=terms&conceptid=fma\%3AImmaterial_anatomical_entity`

```
<class>
          <name>RID13218</name>
          <type>anatomy_metaclass</type>
          <own_slot_value>
                 <slot_reference>FMAID</slot_reference>
                 <value value_type="string">67112</value>
          </own_slot_value>
          <own_slot_value>
                 <slot_reference>Synonym</slot_reference>
                 <value value_type="string">immaterial physical anatomical
                 entity</value>
          </own_slot_value>
          <own_slot_value>
                 <slot_reference>Non-English_name</slot_reference>
                 <value value_type="string">immaterielles körperliches
                 anatomisches Wesen</value>
          </own_slot_value>
          <own_slot_value>
                 <slot_reference>Preferred_name</slot_reference>
                 <value value_type="string">immaterial anatomical entity</value>
          </own_slot_value>
          <own_slot_value>
                 <slot_reference>ORIG_Preferred_Name</slot_reference>
                 <value value_type="string">immaterial anatomical entity</value>
          </own_slot_value>
          <own_slot_value>
                 <slot_reference>Definition</slot_reference>
                 <value value_type="string">Physical anatomical entity which is a
                 three-dimensional space, surface, line or point associated with a
                 material anatomical entity. Examples: body space, surface of heart,
                 costal margin, apex of right lung, anterior compartment of
                 right arm.</value>
          </own_slot_value>
          <own_slot_value>
                 <slot_reference>Is_A</slot_reference>
             4     <value value_type="class">RID13441</value>
          </own_slot_value>
          <own_slot_value>
                 <slot_reference>Has_Subtype</slot_reference>
                 <value value_type="class">RID13221</value>
                 <value value_type="class">RID13250</value>
                 <value value_type="class">RID13291</value>
                 <value value_type="class">RID13307</value>
                 <value value_type="class">RID15845</value>
                 <value value_type="class">RID13217</value>
          </own_slot_value>
          <own_slot_value>
                 <slot_reference>:ROLE</slot_reference>
                 <value value_type="string">Concrete</value>
          </own_slot_value>
          <superclass>RID13441</superclass>
</class>
```

[Example of growing complexity in ontologies by means of a RadLex class.]

It seems that the RadLex ontology in this particular case reuses many elements of FMA, as the focus of RadLex is rather on phenomena that can be observed in correlation with specific organs and not the organs themselves. While this integration of terminological and linguistic knowledge in the field of anatomy is obviously a good move, re-using established terminology, it appears that it could be more beneficial to provide this pool of information independently from the ontologies modeling the domain. Clear links between the original ontology and terms used as well as linguistic data substantially improve the level of

re-usability and readability of semi-structured or definitional natural language expressions across a large number of ontologies (or taxonomies).

3 Related Models

Several approaches and models emphasize the importance of separating conceptual, terminological, and lexical information. Some concentrate on the terminological aspect [6, 9], while others focus on the lexical aspect [10, 4]. Buitelaar et al. [10] propose a model called *LexInfo* and suggest adding lexical, morphosyntactic, and chunking information to the labels of ontology classes. The authors design an OWL representation scheme for this set of linguistic information and its linking to ontology classes. *LexInfo* supports in this among other aspects the ontology-based semantic annotation of text.

The *Terminae* [5] model suggests having two distinct, but interlinked high levels of classes within ontologies: one for the hierarchy of concepts (and associated relations), and one for (a list of) terms that point to the concepts they denote. Thus, the concept level world gets cleaner and, for example, the very cumbersome manner of encoding synonyms and other related terms as it is done in RadLex (see RadLex example above) can be avoided, since synonyms are encoded on the terminological level of the ontology. One major advantage of this approach is that a subset of a terminology can more easily be identified and reused in other (domain) ontologies. Reymont et al. [9] provide an example of the application of *Terminae* in the automotive domain. We note that in *Terminae* the lemma and part-of-speech information is encoded within the term classes.

A third approach, suggesting the merging of *LexInfo* and *Terminae* is CTL [2]. CTL applies the full model of *LexInfo* to each word in a term. Thereby it completely takes lexical information out of the descriptions of both domain and term classes. This leads to three layers of description within the ontology, where a meta-class has three main subclasses describing domain-class, terminology, and linguistic hierarchies. The linguistic layer is based on and extends *LexInfo*. However, CTL neither proposed a formalization nor an implementation, but instead generally described such an approach. Both Terminae and CTL accumulate the different modules (meta-classes) in one ontology, which supports an internal view on the interaction between them, rendering linking of terms to other ontologies more difficult.

Some approaches emphasize the added benefit of a combination of all three modules for specific tasks (e.g. [7]). Bodenreider [7] makes use of existing terminologies, ontologies, and lexicons for text mining in biomedicine. The emphasis here is on already existing not perfectly compatible resources and the specific task of text mining.

All approaches above agree that natural language processing and subsequent linguistic annotation of the terms used in labels are necessary. In order to ensure interoperability and re-usability, we use standardized models. The Terminological Markup Framework (TMF), defined in ISO 16642, ensures the re-usability of terminological data across applications and the TermBase eXchange (TBX)

format of ISO 30042 represents a best practice for the practical exchange of terminology. In line with ISO 704, we take a concept oriented approach towards terminology, defining terminology as concepts and their designations in a specific domain. Consequently, a term is a verbal designation denoting a general concept in a specific domain. The *lemon* model [4] we discuss below proposes a way to obtain the results of natural language processing and annotation in a modular RDF representation.

4 Modularization of Ontology Labels

We propose LabelNet, a model that modularizes each lexical, linguistic, and terminological function related to ontology labels, establishing a net of interlinked terms with highly detailed information at each level. Term entries in a separate OWL-DL encoded TBX- and TMF-compliant terminology relate semantically to corresponding ontology classes or other conceptual elements and represent the terminological information in detail. Each token[5] of every term entry links to a lexical entry, i.e., to a lemma[6], syntactic information, and possible additional resources such as further ontologies. Fig. 1 exemplifies the structure of LabelNet and shows how each of its modules can be interlinked using SKOS. The example data has been taken from an ontology based on the Belgian National Bank (BNB) taxonomy. Time concepts are linked to the W3C time ontology, e.g., "more than one year" is an interval.

The lexical entries are represented by using partially the *lemon* model [4], which is described in the next section. The semantics of the list of tokens contained in a term is established by referring to the ontology elements on the basis of the term ID in the terminological entries.

By separating the several layers into modules we achieve a more complete and highly detailed perspective of ontology labels. The separation of lexical entries and terms into lexicons and terminologies provides a higher degree of re-usability. In addition, it facilitates a number of computations over these labels, such as the usage of a certain lemma in terms pointing to concepts/role IDs.

4.1 The *lemon* Model

lemon provides a model that can encode lexical information, using among others RDF, URIs and linking mechanisms, so that language data can be exchanged for example in the Linguistic Linked Open Data cloud[7]. The model aims at a strict separation of 'world knowledge' (describing domain objects that are

[5] Tokens can be defined as all meaningful elements in a text that result from the process of tokenization, i.e., breaking up text into words, phrases, symbols or other meaningful elements. The ordered collection box in Fig. 1 contains lists of tokens as they appear in the terms used in the exemplified labels.

[6] A lemma represents the canonical form of a set of words called lexemes. For example, *accrue* is the lemma of *accrued, accruing, accrues,* etc.

[7] http://linguistics.okfn.org/resources/llod/

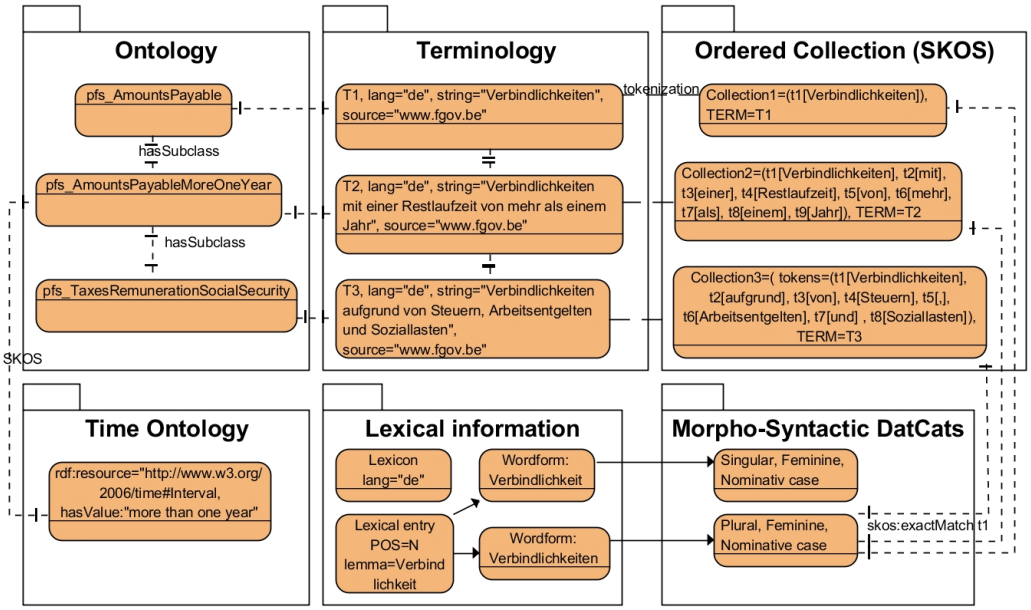

Fig. 1. Simplified example of LabelNet

referenced by lexical objects) from 'word knowledge' (describing lexical objects). It is itself modular, having a core component that can be supplemented with a set of modules to be used, extended, or ignored as required as illustrated in Fig. 2. For example, a morpho-syntactic module can be attached to the core, specifying specific values for words used in the term, such as gender (feminine, masculine, neuter), number (singular, plural, dual) and case (nominative, accusative, etc). As this model in essence enables the creation of a lexicon for a given ontology, it is called an ontology lexicon model. *lemon* as such does not provide an explicit terminological level and refers directly from the lexical entry (in *lemon* a lexical entry represents the whole content of a label) to an ontology element. In contrast, LabelNet stresses the need and the practicability of a terminological level, we re-use only the non-referential part of the *lemon* model.

4.2 Lexicon Module

While *lemon* offers a highly interesting perspective, we think that there are still some shortcomings, or possible improvements. A first case is the fact that lemon supports tokenization of terms included in labels, but not the establishment of the relation between a token represented as a standalone lexical information and the terms in which it can occur. Consequently, we propose an extension that allows for a single lemma to include the information that it is part of a term, in the position specified by the tokenization process. Thereby, the word "Verbindlichkeiten" (German for *amounts payable* or *liabilities*), for example,

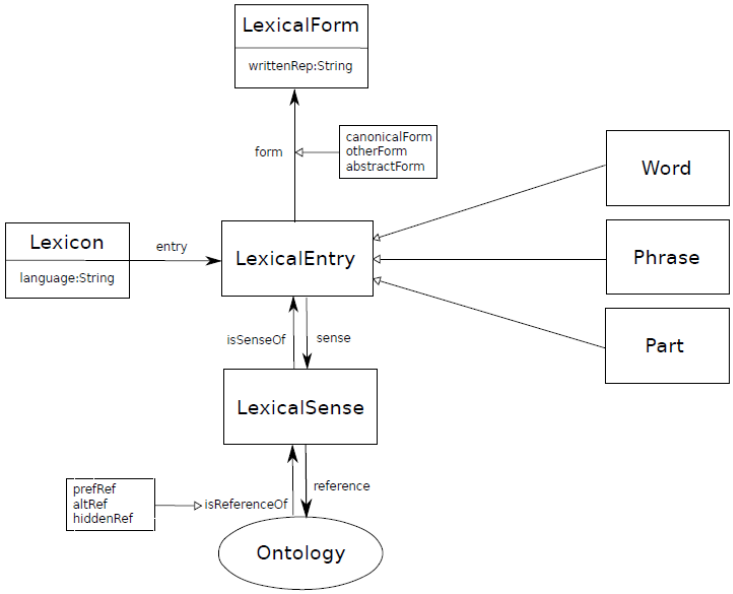

Fig. 2. Simplified representation of the lemon model [4]. The link between lexical and ontology information is established by the reference link

will be linked to a (possibly) substantial number of terms used in various domains (see Fig. 1). In doing so, we can generate a new kind of WordNet, taking into account the inclusion of relevant words in a category of terms. Adopting the idea of *lemon*, we model only lexical and linguistic information in this separate module, linking to semantic values on the basis of the term ID, which itself links to an ontology element.

As a matter of fact, *lemon* entries allow only one semantic reference. The *lemon* model represents the content of labels of one ontology at a time. But frequently one and the same term is used in different (even related) ontologies/taxonomies. In this case, two or more *lemon* entries would be required, leading to redundant lexical/linguistic information only differing in the entry point to elements of different ontologies. One entry pointing to many ontologies represents a more efficient approach. This would also ease generalization over the semantics of such terms.

In case different terms are used in concepts of different ontologies, but a `skos:exactMatch` can be established between these concepts, *lemon* does not provide the means to express the lexical semantic relationship between these terms. As a result, SKOS has to be used as a linking means between those concepts, thereby indirectly establishing the lexical semantic relationship, such as synonymy, between different terms.

Apart from linking different entries or elements of individual modules, certain constraints need to be reflected. For example, in German and English only the

plural of "Verbindlichkeit/liability" might be used within the context of financial reporting. One possibility in *lemon* would be to select only terms in which the word "Verbindlichkeit" appears in its plural form "Verbindlichkeiten". Another possibility, which has our preference, would be to associate a feature structure with the lemma we have extracted from the tokens of the ontology labels, in which additional linguistic information can be encoded. Keeping thus the basic lexicon small, i.e., containing mainly lemmas, and using well-defined feature structures as labels for the edges going from one lemma to a more complex term containing the lemma. We suggest having the constraints expressed in SKOS, linking between a lemma and a term (see Fig. 1):

```
lemma:Verdbindlichkeit -> [plural, feminine, nominative case] -> t1(T3)
```

The above line expresses that only the plural and nominative form of "Verbindlichkeit", which is feminine, can be used in combination with a term (at least the term "T3"') related to a business reporting ontology.

4.3 Terminology in OWL-DL

Terminologies as such consist of terms denominating concepts, their definitions and concept relations. In case of SKOS, these elements are utilized towards building controlled vocabularies, whereas the TermBase eXchange (TBX) format of ISO TC 37 can be described as discourse-oriented terminology [8]. In controlled vocabularies, terms have to be classified as preferred, synonyms being mapped to preferred terms for retrieval purposes. In case of the discourse-related resources, many synonyms are permitted and the attribute "preferred" can be assigned for a prescriptive usage. Wright et al. [8] state that terminologies always relate to special language, "designating multiple preferred terms subject to multiple pragmatic constraints". Thus, the former differs from the latter in that it represents varying conceptual information and semantics with a focus on information retrieval, whereas discourse-oriented terminological resources are more adequate for the purpose at hand.

In our model the terminology is supposed to be reusable for other tasks such as translation, ontology population, ontology building, ontology evolution to name but a few. Instead of using status attributes such as preferred, alternative, and hidden, TBX allows for the use of subset information such as project, application, customer to clarify the difference between synonyms [8].

Terminologies provide greater multiplicity than only `rdfs:labels`. Terms and natural language information acquired for and within the process of ontology building are often lost in the final representation due to a required univocity of each label. Constructing a net of ontology labels and their synonyms acquired in the building of ontologies and extraction of information results in a domain-specific, formalized, and reusable resource for ontologies.

Another reason for transferring natural language information from the ontology to terminologies can be found in its ability to represent conceptual relations

different from ontological relations and thus, enhance the representation of information with linguistic details. For example, a financial reporting ontology classifies *liquid assets* as sibling of *key balance sheet figures*, the latter of which being the parent to *assets*. In contrast, hypernymic relations in the terminology see *assets* as top node, whereas *liquid assets* is one of its children.

TBX is an XML-encoded markup language for the interchange of terminological information. Due to reasons of cardinality and variation its transformation to RDF, i.e., SKOS, turns out to be difficult as described in detail in [8]. Instead of mapping TBX to RDF a member of the OWL family of languages is more adequate to the task. The cardinality of OWL-Lite, however, is restricted to 0 and 1, which in case of many term entries might constitute a problem to be solved with OWL-DL and its ability to allow arbitrary values for cardinality. All core elements of the terminology are children of the top node `owl:datcat` to signify that all subclasses are data categories and interlinked by means of properties such as `unionOf` and `owl:equivalentClass`. A detailed description of rendering TBX in OWL-DL would go beyond the scope of this paper, a representation of terminology in OWL-DL is to be found in [6].

4.4 Step by Step to Modularized Ontologies

Our architectural decisions and selections have been described above, but the specification of the process of obtaining each resource and achieving modularization has yet to be detailed. The main input to building the initial ontology is financial information, such as annual reports of companies, reporting standards (e.g. IFRS, GAAP, XBRL, etc.), stock exchange websites. We extract details from the named sources and build an initial ontology. Furthermore, the extracted information represents the input for the terminology, where all synonyms are depicted. On the basis of the ontology and the terminology, the lexicon is established. So at the core of the following steps lies the formalization of the extracted knowledge in a domain ontology representing our input.

1. Extract labels/terms and linguistic analysis of terms (tokenization, lemmatization, morphological analysis, tagging, parsing, etc).
2. Extract all lemmas, create or map to an existing lemma in a (multilingual) lexicon to collect all lemmas that are used in all possible labels of all possible ontologies.
3. Encode lemmas in *lemon*. Add a data structure on top of each lemma, which lists all the tokens in all labels in which the lemma is reproduced. This linking also reflects the morpho-syntactic features of the token according to its analysis.
4. Record all morpho-syntactic and lexico-syntactic information and patterns in the corresponding addition to the linguistic module.
5. All identical labels are stored as a unique element in a terminology container. Specify term entries as to their conceptual relations and establish proper definitions or adapt definitions existing in the ontology.

6. Each *lemon* represented term is associated with a data structure, i.e., terminology, that points to a variety of ontology elements in which those terms have been introduced.
7. Eliminate all the labels and other linguistic information from the ontology, flattening class entries to domain specific details.

As a result, we have two interlinked ontologies of lemmas and terms as used in ontologies/taxonomies. Thereby, we obtain a subset of language data, which is used in domain ontologies. This can be used in order to analyze textual documents and to annotate them semantically, populate ontologies, or support translations with semantics to name but a few. On the other hand, we have a means for testing ontology mapping or merging.

5 Linking all Modules

The main linking device between ontologies is SKOS, such as the linking between the financial reporting ontology and the time ontology in the example provided in Fig. 1. Especially with multilingual ontologies the individual concepts and their matching by means of SKOS is important. Oftentimes, the pivotal role of English as a source language leads to translations of labels instead of proper localizations. In case of financial reporting standards it is indispensable to take local legal and political regulations affecting the standard into consideration, as the Belgian reporting standard in French might differ substantially from the reporting standard used in France, especially in the use and interpretation of applied French terms.

By conceptualizing the knowledge in each language individually, the ontology is actually created in each language and not simply translated. Thereby, we are in the position of linking for example the English concept `pfs_AmountsPayableMoreOneYear` to the corresponding Italian concept `itcc-ci_DebitiEsigibiliOltreEsercizioSuccessivo` by employing `skos:exactMatch`, which implicitly links the term "Debiti Esigibili Oltre l'Esercizio Successivo" to the English term. For existing monolingual ontologies this proposal might serve as a method for merging several monolingual ontologies by establishing links.

The domain ontology represents the starting point for the linking, containing the initial SKOS links to the terminology, as the terminology might be treated as ontology represented in OWL-DL. From the terminology references to the lexicon holding all individual lemmas can be established. At the same time the terminology represents the interface to lexico- and morpho-syntactic patterns as well as syntactical information as such and all tokens, the result from the process of tokenization.

One part of the linking process is the representation of lexico- and morpho-syntactic patterns and information to support the evolution and extension of existing domain ontologies. Thereby, the construction of new labels is largely facilitated on the basis of the structure of existing labels.

Syntactic information is represented by combining tokens and dependency information of individual terms. Basically, syntactic categories are determined

on the basis of part of speech tagging and phrasal categories are used for syntactic labels. For example N-NP = (length=1, token[1]=N, head=token[1]) represents the term "Verbindlichkeiten", which has the syntactic category "Noun" and the phrasal category "Noun Phrase" with a length of one and token1. For the purpose of standardization, these categories are mainly taken from the ISOcat database[8].

Especially for information extraction in combination with ontology evolution the representation of lexico-syntactic patterns is essential, such as lexico-syntactic ontology design patterns[9] and the famous Hearst patterns. One example for their use is the recognition of relations among entities during information extraction. The following sentence has been taken from the International Financial Reporting Standard (IFRS): "The statement of financial position (sometimes called the balance sheet) includes an entity's assets, liabilities and equity as of the end of the reporting period"[10]. The lexico-syntactic equivalence `<NP class>` call in passive `<NP class>` relation between "statement of financial position" and "balance sheet" enables us to realize that both terms point to the same ontology concept as synonyms, however, including a description of their difference in the definition of the terminology. The Hearst pattern `[NP0] [VBG include] [NP1] [NP2]...` indicates that "assets, liabilities and equity" can be modeled as `subClassOf` "statement of financial position".

6 Conclusion and Future Directions

Modular and encapsulated domain, linguistic, and lexical functions for knowledge modeling enable the support of several IS-related as well as Natural Language Processing (NLP)-driven tasks. Each modularized resource, i.e., ontology, terminology, or lexical information, can either be used as part of the interlinked model we presented or as individual resource for other purposes. One aspect for further improvement certainly is the linking device between the modules, which could be optimized towards an enhanced interoperability with other systems and among the resources themselves.

Acknowledgements. The DFKI part of this work has been supported by the Monnet project (Multilingual ONtologies for NETworked knowledge), co-funded by the European Commission with Grant No. 248458, and by the TrendMiner project, co-funded by the European Commission with Grant No. 287863.

References

1. Aggarwal, N., Wunner, T., Arcan, M., Buitelaar, P., O'Riain, S.: A Similarity Measure based on Semantic, Terminological and Linguistic Information. In: Shvaiko, P., Euzenat, J., Heath, T., Quix, C., Mao, M., Cruz, I.F. (eds.) Proceedings of the 6th International Workshop on Ontology Matching. Bonn, Germany (2011)

[8] http://www.isocat.org/
[9] http://ontologydesignpatterns.org/wiki/Submissions:LexicoSyntacticODPs
[10] http://www.ifrs.org/Home.htm

2. Declerck, T., Lendvai P.: Towards a standardized linguistic annotation of the textual content of labels in knowledge representation systems. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC '10), pp.3836–3839, ELRA, Valetta, Malta (2010)
3. Roche, C., Calberg-Challot, M., Damas, L., Rouard, P.: Ontoterminology: A new paradigm for terminology. In: Dietz, J.L.G. (ed.) International Conference on Knowledge Engineering and Ontology Development. pp. 321–326, Funchal - Madeira, Portugal (2009)
4. McCrae, J., Spohr, D., Cimiano, P.: Linking Lexical Resources and Ontologies on the Semantic Web with Lemon. In: The Semantic Web: Research and Applications. Volume 6643 LNCS, pp. 245-259. Springer, Berlin, Germany (2011)
5. Aussenac-Gilles, N., Szulman, S., Despres, S.: The Terminae Method and Platform for Ontology Engineering from Texts. In: Proceedings of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge. IOS Press, pp. 199–223, (2008)
6. Reymonet A., Thomas, J., Aussenac-Gilles, N.: Modelling ontological and terminological resources in OWL-DL. In:Buitelaar, P., Choi, K.S., Gangemi, A., Huang, C.R (eds) OntoLex 2007, ISWC Workshop. Busan, South-Korea (2007)
7. Bodenreider, O.: Lexical, terminological and ontological resources for biological text mining. In: Ananiadou, S., McNaught, J. (eds) Text mining for biology and biomedicine, p. 43-66, Artech House, London, England (2006)
8. Wright, S. E., Summers, D.: Crosswalking from Terminology to Terminology: Leveraging Semantic Information across Communities of Practice. In: Witt, A., Sasaki, F., Teich, E., Calzolari, N., Wittenburg, P. (eds) Uses and usage of language resource-related standards, LREC, Marrakech, Morocco (2008)
9. Reymonet, A., Thomas, J., Aussenac-Gilles, N.: Ontology based information retrieval: an application to automotive diagnosis. In: Nyberg, M., Frisk, E.m Krisander, M., Aslund, J. (eds) International Workshop on Principles of Diagnosis, pp.9-14, Stockholm, Sweden (2009)
10. Buitelaar, P., Cimiano, P. Haase, P., Sintek, M.: Towards linguistically grounded ontologies. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Bontas Simpler, E.P. (eds) ESWC 2009. pp. 111–125, Springer Berlin/Heidelberg, Heraklion, Crete, Greece (2009)

Syntactic vs. Semantic Locality: How Good Is a Cheap Approximation?

Chiara Del Vescovo[1], Pavel Klinov[2], Bijan Parsia[1], Uli Sattler[1], Thomas Schneider[3], and Dmitry Tsarkov[1]

[1] University of Manchester, UK
{delvescc,bparsia,sattler,tsarkov}@cs.man.ac.uk
[2] University of Ulm, Germany
pavel.klinov@uni-ulm.de
[3] Universität Bremen, Germany
tschneider@informatik.uni-bremen.de

Abstract Extracting a subset of a given OWL ontology that captures all the ontology's knowledge about a specified set of terms is a well-understood task. This task can be based, for instance, on locality-based modules (LBMs). These come in two flavours, syntactic and semantic, and a syntactic LBM is known to contain the corresponding semantic LBM. For syntactic LBMs, polynomial extraction algorithms are known, implemented in the OWL API, and being used. In contrast, extracting semantic LBMs involves reasoning, which is intractable for OWL 2 DL, and these algorithms had not been implemented yet for expressive ontology languages.
We present the first implementation of semantic LBMs and report on experiments that compare them with syntactic LBMs extracted from real-life ontologies. Our study reveals whether semantic LBMs are worth the additional extraction effort, compared with syntactic LBMs.

1 Introduction

Extracting a subset of a given OWL ontology that captures all the ontology's knowledge about a specified set of concept and role names is an interesting task for various applications, and it is by now well-understood [2,10,11]. In general, we consider a setting where, for a given *signature*, we want to determine a (small) subset of a given ontology such that any axiom over the signature entailed by the ontology is also entailed by the subset. For expressive logics, this task can be implemented by making use of the notion of *locality*, and results in what is known as locality-based modules (LBMs) [2]. Locality comes in many different flavours, in particular there are notions of syntactic and semantic locality. A syntactic LBM is known to contain the corresponding semantic LBM, but might also contain extra axioms which are, because they are not in the semantic LBM, superfluous for entailments over the given signature. Algorithms for the extraction of syntactic LBMs are known that run in time that is polynomial in the size of the ontology (thus much cheaper than reasoning), implemented in the OWL

API, and being used. In contrast, despite the fact that algorithms for extracting semantic LBMs are known, until now and to the best of our knowledge, they had not yet been implemented. Moreover, these involve entailment checking, and are thus intractable for expressive profiles of OWL 2.

We present the first implementation of semantic LBMs and report on experiments that compare them with syntactic LBMs extracted from real-life ontologies. The contributions of this paper are as follows: we show with statistical significance that, for almost all members of a large corpus of existing ontologies, there is no difference between any syntactic LBM and its corresponding semantic LBM. In the few cases where differences occur, these differences are modest and not worth the increased computation time needed to compute semantic LBMs. In addition, we isolate two types of axioms that lead to differences, where one is a simple tautology that can, in principle, be detected by a straightforward addition to the syntactic locality checker. Furthermore, our results show that the extraction of semantic LBMs, which is in principle hard, seems feasible in practice. The lesson we learn from these results is that "Cheap is Great"!

2 Preliminaries

We assume the reader to be familiar with OWL and the underlying description logic $\mathcal{SROIQ}$ [1,8], and will define the central notions around locality-based modularity [2].

Let $\mathsf{N_C}$ be a set of concept names, and $\mathsf{N_R}$ a set of role names. A *signature* Σ is a set of *terms*, i.e., a set $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$ of concept and role names. We can think of a signature as specifying a topic of interest. Axioms that only use terms from Σ can be thought of as "on-topic", and all other axioms as "off-topic". For instance, if $\Sigma = \{\mathsf{Animal}, \mathsf{Duck}, \mathsf{Grass}, \mathsf{eats}\}$, then $\mathsf{Duck} \sqsubseteq \exists \mathsf{eats}.\mathsf{Grass}$ is on-topic, while $\mathsf{Duck} \sqsubseteq \mathsf{Bird}$ is off-topic.

Any concept, role, or axiom that uses only terms from Σ is called a *Σ-concept*, *Σ-role*, or *Σ-axiom*. Given any such object X, we call the set of terms in X the *signature of* X and denote it with $\widetilde{X}$.

Given an interpretation $\mathcal{I}$, we denote its restriction to the terms in a signature Σ with $\mathcal{I}|_\Sigma$. Two interpretations $\mathcal{I}$ and $\mathcal{J}$ are said to *coincide on a signature* Σ, in symbols $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$, if $\Delta^\mathcal{I} = \Delta^\mathcal{J}$ and $X^\mathcal{I} = X^\mathcal{J}$ for all $X \in \Sigma$.

There are a number of variants of the notion of conservative extensions, which capture the desired preservation of knowledge to different degrees. We focus on the deductive variant.

Definition 1. Let $\mathcal{M} \subseteq \mathcal{O}$ be $\mathcal{SROIQ}$-ontologies and Σ a signature.

(1) $\mathcal{O}$ is a *deductive Σ-conservative extension* (Σ-*dCE*) of $\mathcal{M}$ if, for all $\mathcal{SROIQ}$-axioms α with $\widetilde{\alpha} \subseteq \Sigma$, it holds that $\mathcal{M} \models \alpha$ if and only if $\mathcal{O} \models \alpha$.
(2) $\mathcal{M}$ is a *dCE-based module for* Σ of $\mathcal{O}$ if $\mathcal{O}$ is a Σ-dCE of $\mathcal{M}$.

Unfortunately, deciding in general if a set of axioms is a module in this sense is hard or even impossible for expressive DLs [6,12], and finding a minimal one

is even more so. However, "good sized" modules that are efficiently computable have been introduced [2]. They are based on the *locality* of single axioms, which means that, given Σ, the axiom can always be satisfied independently of the interpretation of the Σ-terms, but in a restricted way: by interpreting all non-Σ terms either as the empty set ($\emptyset$-locality) or as the full domain[4] (Δ-locality).

Definition 2. A $\mathcal{SROIQ}$-axiom α is called *$\emptyset$-local (Δ-local) w.r.t. signature Σ* if, for each interpretation $\mathcal{I}$, there exists an interpretation $\mathcal{J}$ such that $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$, $\mathcal{J} \models \alpha$, and for each $X \in \widetilde{\alpha} \setminus \Sigma$, $X^\mathcal{J} = \emptyset$ (for each $C \in \widetilde{\alpha} \setminus \Sigma$, $C^\mathcal{J} = \Delta$ and for each $R \in \widetilde{\alpha} \setminus \Sigma$, $R^\mathcal{J} = \Delta \times \Delta$).

It has been shown in [2] that $\mathcal{M} \subseteq \mathcal{O}$ and all axioms in $\mathcal{O} \setminus \mathcal{M}$ being $\emptyset$-local (or all axioms being Δ-local) w.r.t. $\Sigma \cup \widetilde{\mathcal{M}}$ is sufficient for $\mathcal{O}$ to be a Σ-dCE of $\mathcal{M}$. The converse does not hold: e.g., the axiom $A \equiv B$ is neither $\emptyset$- nor Δ-local w.r.t. $\{A\}$, but the ontology $\{A \equiv B\}$ is an $\{A\}$-dCE of the empty ontology.

Furthermore, locality can be tested using available DL-reasoners [2], which makes this problem considerably easier than testing conservativity. However, reasoning in expressive DLs is still complex, e.g. N2ExpTime-complete for $\mathcal{SROIQ}$ [9]. In order to achieve *tractable* module extraction, a syntactic approximation of locality has been introduced in [2]. The following definition captures only the case of $\mathcal{SHQ}$-TBoxes and can straightforwardly be extended to $\mathcal{SROIQ}$ ontologies.

Definition 3. An axiom α is called *syntactically $\bot$-local* ($\top$-*local*) *w.r.t. signature Σ* if it is of the form $C^\bot \sqsubseteq C$, $C \sqsubseteq C^\top$, $C^\bot \equiv C^\bot$, $C^\top \equiv C^\top$, $R^\bot \sqsubseteq R$ ($R \sqsubseteq R^\top$), or $\mathsf{Trans}(R^\bot)$ ($\mathsf{Trans}(R^\top)$), where C is an arbitrary concept, R is an arbitrary role name, $R^\bot \notin \Sigma$ ($R^\top \notin \Sigma$), and $C^\bot$ and $C^\top$ are from $\mathrm{Bot}(\Sigma)$ and $\mathrm{Top}(\Sigma)$ as defined in Part (a) (resp. (b)) of the table below.

(a) $\bot$-*Locality*	Let $A^\bot, R^\bot \notin \Sigma$, $C^\bot \in \mathrm{Bot}(\Sigma)$, $C^\top_{(i)} \in \mathrm{Top}(\Sigma)$, $\bar{n} \in \mathbb{N} \setminus \{0\}$
$\mathrm{Bot}(\Sigma)$	$::= A^\bot \mid \bot \mid \neg C^\top \mid C \sqcap C^\bot \mid C^\bot \sqcap C \mid \exists R.C^\bot \mid \geqslant \bar{n}\, R.C^\bot \mid \exists R^\bot.C \mid \geqslant \bar{n}\, R^\bot.C$
$\mathrm{Top}(\Sigma)$	$::= \top \mid \neg C^\bot \mid C^\top_1 \sqcap C^\top_2 \mid \geqslant 0\, R.C$

(b) $\top$-*Locality*	Let $A^\top, R^\top \notin \Sigma$, $C^\bot \in \mathrm{Bot}(\Sigma)$, $C^\top_{(i)} \in \mathrm{Top}(\Sigma)$, $\bar{n} \in \mathbb{N} \setminus \{0\}$
$\mathrm{Bot}(\Sigma)$	$::= \bot \mid \neg C^\top \mid C \sqcap C^\bot \mid C^\bot \sqcap C \mid \exists R.C^\bot \mid \geqslant \bar{n}\, R.C^\bot$
$\mathrm{Top}(\Sigma)$	$::= A^\top \mid \top \mid \neg C^\bot \mid C^\top_1 \sqcap C^\top_2 \mid \exists R^\top.C^\top \mid \geqslant \bar{n}\, R^\top.C^\top \mid \geqslant 0\, R.C$

It has been shown in [2] that $\bot$-locality ($\top$-locality) of an axiom α w.r.t. Σ implies $\emptyset$-locality (Δ-locality) of α w.r.t. Σ. Therefore, all axioms in $\mathcal{O} \setminus \mathcal{M}$ being $\bot$-local (or all axioms being $\top$-local) w.r.t. $\Sigma \cup \widetilde{\mathcal{M}}$ is sufficient for $\mathcal{O}$ to be a Σ-dCE of $\mathcal{M}$. The converse does not hold; examples can be found in [2].

For each of the four locality notions, modules of $\mathcal{O}$ are obtained by starting with an empty set of axioms and subsequently adding axioms from $\mathcal{O}$ that are Σ-non-local. In order for this procedure to be correct, the signature against which

[4] Or, in the case of roles, the set of all pairs of domain elements.

locality is checked has to be extended with the terms in the axioms that are added in each step, so that the resulting module $\mathcal{M}$ consists of all the non-local axioms with respect to $\Sigma \cup \widetilde{\mathcal{M}}$. Definition 4 (1) introduces locality-based modules, which are always dCE-based modules [2], although not necessarily minimal ones. Modules based on syntactic (semantic) locality can be made smaller by iteratively nesting $\top$- and $\bot$-extraction (Δ- and $\emptyset$-extraction), and the result is still a dCE-based module [2,13]. These so-called $\top\bot^*$-modules ($\Delta\emptyset^*$-modules) are introduced in Definition 4 (3).

Definition 4. Let $x \in \{\emptyset, \Delta, \bot, \top\}$, $yz \in \{\top\bot, \Delta\emptyset\}$, $\mathcal{O}$ an ontology and Σ a signature.

(1) An ontology $\mathcal{M}$ is the *x-module of $\mathcal{O}$ w.r.t. Σ* if it is the output of Algorithm 1. We write $\mathcal{M} = x\text{-mod}(\Sigma, \mathcal{O})$.
(2) An ontology $\mathcal{M}$ is the *yz-module of $\mathcal{O}$ w.r.t. Σ*, written $\mathcal{M} = yz\text{-mod}(\Sigma, \mathcal{O})$, if $\mathcal{M} = y\text{-mod}(\Sigma, z\text{-mod}(\Sigma, \mathcal{O}))$.
(3) Let $(\mathcal{M}_i)_{i \geqslant 0}$ be a sequence of ontologies such that $\mathcal{M}_0 = \mathcal{O}$ and $\mathcal{M}_{i+1} = yz\text{-mod}(\Sigma, \mathcal{M}_i)$ for every $i \geqslant 0$. For the smallest $n \geqslant 0$ with $\mathcal{M}_n = \mathcal{M}_{n+1}$, we call $\mathcal{M}_n$ the *yz^*-module of $\mathcal{O}$ w.r.t. Σ*, written $\mathcal{M} = yz^*\text{-mod}(\Sigma, \mathcal{O})$.

Algorithm 1 Extract a locality-based module

Input: Ont. $\mathcal{O}$, sig. Σ, $x \in \{\emptyset, \Delta, \bot, \top\}$ **Output:** x-module $\mathcal{M}$ of $\mathcal{O}$ w.r.t. Σ

$M \leftarrow \emptyset$; $\mathcal{O}' \leftarrow \mathcal{O}$
repeat
  changed $\leftarrow$ `false`
  for all $\alpha \in \mathcal{O}'$ **do**
    if α not x-local w.r.t. $\Sigma \cup \widetilde{\mathcal{M}}$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{\alpha\}$; $\mathcal{O}' \leftarrow \mathcal{O}' \setminus \{\alpha\}$; changed $\leftarrow$ `true`
until changed = `false`
return $\mathcal{M}$

As for (1), it has been shown in [2] that the output $\mathcal{M}$ of Algorithm 1 does not depend on the order in which the axioms α are selected.[5] Furthermore, the integer n in (3) exists because the sequence $(\mathcal{M}_i)_{i \geqslant 0}$ is decreasing (more precisely, we have $\mathcal{M}_0 \supset \cdots \supset \mathcal{M}_n = \mathcal{M}_{n+1} = \ldots$). Due to monotonicity properties of locality-based modules, the dual notions of $\bot\top^*$- and $\emptyset\Delta^*$-modules are uninteresting because they coincide with those of $\top\bot^*$- and $\Delta\emptyset^*$-modules.

Roughly speaking, a Δ- or $\top$-module for Σ gives a view from above because it contains all subclasses of class names in Σ, while a $\emptyset$- or $\bot$-module for Σ gives a view from below since it contains all superconcepts of concept names in Σ.

Modulo the locality check, Algorithm 1 runs in time cubic in $|\mathcal{O}| + |\Sigma|$ [2]. Modules based on $\bot/\top$-locality are therefore a feasible approximation for modules based on $\emptyset/\Delta$-locality. In both cases, modules are extracted axiom by axiom

[5] Our algorithm is a special case of the one in [2, Figure 4].

but, as said above, the $\emptyset/\Delta$-locality check is more complex. A module extractor is implemented in the OWL API[6] and SSWAP[7]. To summarize:

1. Given an ontology $\mathcal{O}$, the semantic module $\mathcal{M}^{\text{sem}}_{\Sigma}$ for a signature Σ is contained in the corresponding syntactic module $\mathcal{M}^{\text{syn}}_{\Sigma}$ for the same seed signature.[8] This means that in principle more unnecessary axioms for preserving entailments over Σ can end up in syntactic modules rather than in semantic modules.
2. The extraction of a syntactic module can be done in polynomial time w.r.t. the size of the ontology $\mathcal{O}$. In contrast, the extraction of a semantic module is as hard as reasoning.

3 Experimental design

The main aim of this paper is to investigate how well syntactic locality approximates semantic locality. In particular, we want to see how (un)likely it is that syntactic locality-based modules are larger than semantic locality-based ones and how large these differences are. We also want to understand empirically how much more costly semantic locality is in terms of performance.

Selection of the Corpus. For our experiments, we have built a corpus containing: (1) from the TONES repository,[9] those ontologies that have already been studied in a previous work on modularity [4]: Koala, Mereology, University, People, mini-Tambis, OWL-S, Tambis, Galen; (2) all ontologies from the NCBO BioPortal ontology repository.[10]

We then filter out all those the ontologies for which at least one of the following problems occurs: the ontology is impossible to download; the .owl file is corrupted when downloaded; the file is not parseable; the ontology is inconsistent. Furthermore, due to time constraints, we exclude from this preliminary investigation all ontologies whose size exceeds $10,000$ axioms.

This selection results in a corpus of 156 ontologies, which greatly differ in size and expressivity [7], as summarized in Table 3. For a full list of the corpus, please refer to the technical report: `http://arxiv.org/abs/1207.1641`

Repository	Range of expressivity	Range #axs.	Range sig. size
BioPortal	$\mathcal{ALCN}$-$\mathcal{SHIN}(\mathcal{D})/\mathcal{SOIN}(\mathcal{D})$	38–4,735	21–3,161
TONES	$\mathcal{AL}$-$\mathcal{SROIF}(\mathcal{D})/\mathcal{SHOIQ}(\mathcal{D})$	13–9,629	14–9,221

Table 1. Ontology corpus

[6] `http://owlapi.sourceforge.net`
[7] `http://sswap.info`
[8] Recall that $\bot$-syntactic modules approximate $\emptyset$-semantic modules, while $\top$-syntactic modules approximate Δ-semantic modules.
[9] `http://owl.cs.manchester.ac.uk/repository/`
[10] `http://bioportal.bioontology.org`

Comparing Syntactic and Semantic Locality. In order to compare syntactic and semantic locality, we want to understand:

1. whether, for a given seed signature Σ, the semantic Σ-module is likely to be smaller than the syntactic Σ-module, and if so by how much,[11]
2. how feasible the extraction of semantic modules is.

Here, we focus on the two corresponding notions of $\emptyset$-semantic locality and $\bot$-syntactic locality. In particular, $\bot$-syntactic locality has been throughly investigated in previous work [3], and it has proven to have many interesting properties. A completion of the investigation described in this paper for all fundamental notions of modules is planned in our future work.

Due to the recursive nature of the locality-based module extraction algorithm, we want to investigate locality both on a

- per-axiom basis: given an axiom α and a signature Σ, is it likely that α is semantically $\emptyset$-local w.r.t. Σ but not syntactically $\bot$- local w.r.t. Σ?
- per-module basis: given a signature Σ, is it likely that $\bot\text{-mod}(\Sigma, \mathcal{O}) \neq \emptyset\text{-mod}(\Sigma, \mathcal{O})$? If yes, is it likely that the difference is large?

Hence we need to pick, for each ontology in our corpus, a suitable set of signatures, and this poses a significant problem. First, we do not yet have enough insight into what typical seed signatures are for module extraction. One could assume that large ones are rarely relevant for module extraction—why bother with extracting a large module—but this still leaves a large, i.e., exponential space of possible seed signatures. If $m = \#\widetilde{\mathcal{O}}$, there are 2^m possible seed signatures for which axioms can be tested for locality and for which modules can be extracted. Hence a full investigation is infeasible.

One could assume that the comparison between semantic and syntactic modules could be easier since many signatures can lead to the same module. In other words, the statistically significant number of modules w.r.t. the total number of modules is not larger than that of seed signatures needed w.r.t. the total number of seed signatures. In previous work [4,5], however, modules have been studied with respect to how numerous they are in real-world ontologies. The experiments carried out suggest that the number of modules in ontologies is, in general, exponential w.r.t. the size of the ontology. Moreover, the extraction of enough *different* modules can be hard, because by looking just at seed signatures there is no chance to avoid the extraction of the same module many times. In particular, for a module $\mathcal{M}$ there can be exponentially many seed signatures w.r.t. $\#\widetilde{\mathcal{M}}$ that generate $\mathcal{M}$ [3].

As a consequence, we compare the two kinds of locality of axioms—both on a per-axiom basis and a per-module basis—w.r.t. random signatures. To avoid any bias, we select a random signature as follows: we set each named entity `E` in the ontology to have probability $p = 1/2$ of being included in the signature. Thus each seed signature has the same probability to be chosen. For ontologies whose signature exceeds 9 entities, in order to get results where the

[11] Recall that the semantic Σ-module is always a subset of the syntactic Σ-module.

true proportion of differences between the two notions of locality lies in the confidence interval (±5%) with confidence level 95%, we have to select only 400 random signatures [14]. That is, we need to test only 400 random signatures to have a confidence of 95% (±5%) that the differences/equalities we observe reflect the real ones.

Non-random seed signatures. A module, in general, does not necessarily show any internal coherence: intuitively, if we had an ontology describing some knowledge from both the domains of Geology and of Philosophy, we could still extract the module for the signature $\Sigma = \{\texttt{Epistemology}, \texttt{Mineral}\}$. This module is likely to be the union of the two disjoint modules for $\Sigma_1 = \{\texttt{Epistemology}\}$ and $\Sigma_2 = \{\texttt{Mineral}\}$. This combinatorial behaviour can lead to exponentially many modules in the size of the signature of the ontology and indeed, as mentioned above, the number of modules in ontologies seems to be exponential [4,5].

In contrast to *general* modules, *genuine modules* can be called coherent: they are defined as those modules that cannot be decomposed into the union of two different modules. Notably, there are only linearly many genuine modules in the size of the ontology $\mathcal{O}$, and the set of genuine modules is a base for all general modules: any module is either genuine or the union of genuine modules. The linear bound on the number of genuine modules is due to the fact that, for each genuine x-module $\mathcal{M}$, there is an axiom α such that $\mathcal{M} = x\text{-mod}(\tilde{\alpha}, \mathcal{O})$.

Thus genuine modules can be said to be interesting modules that we can fully investigate. Hence in addition to the above mentioned investigation of $\bot$- and $\emptyset$-modules for random signatures, we also look at all axiom signatures.

In summary, we test:

(T1) for random seed signatures Σ,
- **(a)** for each axiom α in our corpus, is α semantically $\emptyset$-local w.r.t. Σ but not syntactically $\bot$- local w.r.t. Σ?
- **(b)** is $\bot\text{-mod}(\Sigma, \mathcal{O}) \neq \emptyset\text{-mod}(\Sigma, \mathcal{O})$? If yes, we determine the difference and its size.

(T2) for each axiom signature from our corpus, is $\bot\text{-mod}(\tilde{\alpha}, \mathcal{O}) \neq \emptyset\text{-mod}(\tilde{\alpha}, \mathcal{O})$? If yes, we determine the difference and its size.

4 Experimental comparison

No differences. The main result of the experiment is that, for 151 of the 156 ontologies we tested, no difference between $\bot$- and $\emptyset$-locality can be observed. These 151 ontologies exclude the two NCBO BioPortal ontologies EFO (Experimental Factor Ontology) and SWO (Software Ontology), as well as Koala, miniTambis, and Tambis. More specifically, for every generated seed signature, the corresponding $\bot$- and $\emptyset$-module agree, and every axiom is either $\bot$- and $\emptyset$-local, or neither. This statement applies to all randomly generated seed signatures as well as for *all* axiom signatures – which are seed signatures for all genuine modules. We can therefore draw the following conclusions for the 151 ontologies with respect to (T1) and (T2) above.

(T1) Given an arbitrary seed signature Σ, there is no difference (a) between $\bot$- and $\emptyset$-locality of any given axiom w.r.t. Σ and (b) between the $\bot$- and $\emptyset$-modules for Σ, both times at a significance level of 0.05.

(T2) Given *any* axiom signature Σ, there is no difference between the $\bot$- and $\emptyset$-modules for Σ.

In the case of the 151 ontologies, the extraction of a $\emptyset$-module (with tautology tests performed by FaCT++) often took considerably longer than the extraction of the corresponding $\bot$-module. For example, for MoleculeRole, the largest of the 151 ontologies, times to extract a $\bot$-module (test all axioms for $\bot$-locality, respectively) ranged between 27 and 169ms (21 and 77ms, respectively), while the extraction of a $\emptyset$-module (test of all axioms for $\emptyset$-locality, resp.) took up to $6\times$ as long, on average $2.7\times$ ($2.0\times$, resp.). It is also worth noting that the ontologies Galen and People, which are renowned for having particularly large $\bot$-modules [2,5], are among those without differences between $\bot$- and $\emptyset$-locality.

Differences. For the five ontologies where differences between $\bot$- and $\emptyset$-modules (or -locality) occur, we isolated two types of culprits – axioms which are not $\bot$-local w.r.t. some signature Σ, but which are $\emptyset$-local w.r.t. Σ. Type-1 culprits are simple tautologies that have accidentally entered the "inferred view" – i.e., closure under certain entailments – of two ontologies. They do not occur in the original "asserted" versions and can, in principle, be detected by a slightly refined syntactic locality check. Type-2 culprits are definitions of concept names via a conjunction that satisfies certain conditions explained below. There are not many type-1 and type-2 axioms in the affected ontologies, and the observed differences are comparably small. Table 2 gives an overview of the differences observed.

Type-1 culprits are axioms `InverseObjectProperties(P, InverseOf(P))`, where `P` is a role. This translates into the tautology $\mathtt{P} \equiv (\mathtt{P}^-)^-$ in DL notation. Such an axiom is therefore $\emptyset$-local w.r.t. any signature. However, it behaves differently for $\bot$-locality: if the signature Σ contains `P`, then both sides of the equation are neither in $\mathrm{Bot}(\Sigma)$ nor in $\mathrm{Top}(\Sigma)$, hence the axiom is considered non-local; otherwise, both sides are $\bot$-equivalent, hence the axiom is local.

Type-1 axioms occur in the "inferred view" of the ontologies EFO and SWO. Table 2 shows the relatively modest differences caused by these axioms. In all cases, there are no other axioms in the differences. This means that no differences occur for the non-inferred original versions of EFO and SWO.

Type-2 culprits are complex definitions $A \equiv C$ of a concept name A where C is a disjunction that contains both a universal and an existential (or minimum cardinality) restriction on the same role. This affects the ontologies Koala, miniTambis, and Tambis. The effect is best illustrated for Koala, which contains exactly one such axiom, namely $\mathtt{M} \equiv \mathtt{S} \sqcap \forall \mathtt{c}.\mathtt{F} \sqcap \forall \mathtt{g}.\{\mathtt{m}\} \sqcap =3\,\mathtt{c}.\top$, where we have abbreviated the concept names `MaleStudentWith3Daughters`, `Student`, `Female`, the roles `hasChildren`, `hasGender`, and the nominal `male`. Now if the signature against which the axiom is tested for locality contains $\{\mathtt{S}, \mathtt{c}, \mathtt{g}\}$ but

Ontology	#axs	#differences		difference sizes #axs	rel.	time ratio avg.	culprit type and frequency
SWO	3446	T1 a	400	6–22	0–1%	3.31	1 (30×)
		T1 b	400	23–29	1–2%	5.11	
		T2	3446	3–1	1–5%	5.86	
EFO	6008	T1 a	400	8–24	0–1%	1.42	1 (32×)
		T1 b	400	13–30	0–1%	1.38	
		T2	128	1–4	9–17%	—	
Koala	42	T1 a	0	0	0%	—	2 (1×)
		T1 b	2	1	3%	—	
		T2	0	0	0%	—	
miniTambis	170	T1 a	68	1–2	1–3%	—	2 (3×)
		T1 b	93	1–4	1–3%	—	
		T2	26	1–7	6–75%	—	
Tambis	592	T1 a	58	1–3	0–1%	3.31	2 (11×)
		T1 b	229	2–11	0–2%	5.01	
		T2	191	4–41	2–26%	—	

Table 2. Overview table of differences observed. The columns show: the ontology name; the overall number of axioms; the name of the test (see list on Page 7); the number of cases with differences; the number of axioms in the differences (absolute and relative to the $\bot$-case); the average time ratio $\emptyset : \bot$ ("—" indicates that no reliable statement is possible: the time for $\bot$ is only a few, often 0, milliseconds); the type of culprit present and the number of axioms of this type.

neither `M` nor `F`, then this axiom is not $\bot$-local because none of the conjuncts on the right-hand side is in $\mathrm{Bot}(\Sigma)$. On the other hand, this axiom is a tautology when `M` and `F` are replaced by $\bot$: the conjunction $\forall \mathsf{c}.\bot \sqcap =3\,\mathsf{c}.\top$ cannot have any instances, regardless of how `c` is interpreted.

For Koala, this effect only causes two singleton differences between sets of local axioms for the randomly generated seed signatures, as shown in Table 2. For axiom signatures, there is no difference. Interestingly, this effect does not propagate to modules: for all signatures, $\bot$- and $\emptyset$-modules are the same. The reason might be that (a) `g` is used in many axioms and is thus very likely to contribute to the extended signature during module extraction, and (b) then the axiom defining `F` is no longer local, which "pulls" `F` into the extended signature, preventing the observed effect.

In miniTambis and Tambis, this effect is much stronger and affects a large proportion of modules, as shown in Table 2. The differences in these cases do not only consist of culprit axioms, but also of axioms that become non-local after the signature has been extended by the terms in the culprit axioms. Still, the size of the differences is mostly modest while, for Tambis, the $\emptyset$-locality test ($\emptyset$-module extraction) takes on average over three times (five times) as long as the $\bot$-locality test ($\bot$-module extraction).

5 Conclusion and Outlook

Summary. We obtain two main observations from the experiments carried out.

- In practice, there is no or little difference between semantic and syntactic locality. That is, the computationally cheaper syntactic locality is a good approximation of semantic locality.
- Though in principle hard to compute, semantic modules can be extracted rather fast in practice.

These results suggest that it is questionable to conclude that semantic locality should be preferred to syntactic locality. In terms of computation time, there is often a benefit in using syntactic locality: the average speed-up compared to the extraction of a semantic-locality based module is by a factor of up to 6. For some particular module pairs, it is higher by an order of magnitude. The gain in module size is zero or so small that it is hard to justify the extra time spent. In particular, there is no gain in size for the ontologies Galen and People, which are "renowned" for having disproportionately large modules [2,5].

Our results are interesting not only because they provide an evaluation of how good the cheap syntactic locality approximates semantic locality, but also because they enabled us to fix bugs in the implementation of syntactic modularity. For example, earlier data from the experiment have shown that reflexivity axioms had been treated incorrectly by the syntactic locality checker.

Future Work. It is evident that this work is preliminary. It investigates only the differences between the related notions of $\bot$- and $\emptyset$-locality. We plan to extend the same study to other notions of locality, in particular, nested modules ($\top\bot^*$- vs. $\Delta\emptyset^*$-modules) – these notions are the most economical in terms of module size. Moreover, we want to extend the investigation to the remaining larger ontologies in the BioPortal repository and further large ontologies, e.g., some versions of the NCI Thesaurus[12]. Preliminary results with a version that is not among the regular releases show differences due to type-2 culprits, but we have not included them here because the differences disappear after removing axioms that were introduced due a problem with object and annotation properties when the ontology file is parsed by the OWL API. This behaviour is yet to be investigated and explained.

Another interesting extension is to modify the seed signature sampling. Currently, the random variable "size of the seed signature generated" follows the binomial distribution with expected value $m/2$ and variance $m/4$. Hence, most signatures in the sample have size around $m/2$; small and large signatures are underrepresented. For example, for one ontology with 915 terms, all signature sizes lay between 422 and 509. One might argue that, for big ontologies, the typical module extraction scenario does not require large seed signatures – but it does sometimes require relatively small seed signatures, for example, when a module is extracted to efficiently answer a given entailment query of typically small size.

[12] Downloadable from `http://evs.nci.nih.gov/ftp1/NCI_Thesaurus`

On the other hand, large modules resulting from larger seed signatures may be more likely to differ. We therefore plan an alternative seed signature sampling via bins for average signature sizes: repeat the current sampling procedure scaled to several subintervals of the range of possible signature sizes.

Our current results answer the question whether there is a significant difference between the two locality notions *with respect to a given signature*. It is also interesting to ask the same question relative to a given module. To answer it, the sampling of modules instead of seed signatures requires further investigation.

Acknowledgment. We thank Rafael Gonçalves and the anonymous reviewers for helpful comments.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artif. Intell. Research 31, 273–318 (2008)
3. Del Vescovo, C., Gessler, D., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Winget, A.: Decomposition and Modular Structure of BioPortal Ontologies. In: Proc. ISWC-11 (2011)
4. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: an empirical study. In: Proc. of WoMO-10. Frontiers in AI and Appl., vol. 211, pp. 11–24. IOS Press (2010)
5. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: atomic decomposition and module count. In: Proc. of WoMO-11. Frontiers in AI and Appl., vol. 230, pp. 25–39. IOS Press (2011)
6. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Proc. of KR-06. pp. 187–197 (2006)
7. Horridge, M., Parsia, B., Sattler, U.: The state of bio-medical ontologies. In: Proc. of 2011 ISMB Bio-Ontologies SIG (2011)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR-06. pp. 57–67 (2006)
9. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Proc. of KR-08. pp. 274–284 (2008)
10. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic modularity and module extraction in description logics. In: Proc. of ECAI-08. Frontiers in AI and Appl., vol. 178, pp. 55–59. IOS Press (2008)
11. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to DL-Lite. Artificial Intelligence 174(15), 1093–1141 (2010)
12. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of IJCAI-07. pp. 453–458 (2007)
13. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Proc. of DL 2009. `ceur-ws.org`, vol. 477 (2009)
14. Smithson, M.: Confidence Intervals. Quantitative Applications in the Social Sciences, Sage Publications (2003)

LoLa: A Modular Ontology of Logics, Languages, and Translations

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

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

Abstract. The Distributed Ontology Language (DOL), currently being standardised within the OntoIOp (Ontology Integration and Interoperability) activity of ISO/TC 37/SC 3, aims at providing a unified framework for (1) ontologies formalised in heterogeneous logics, (2) modular ontologies, (3) links between ontologies, and (4) annotation of ontologies. This paper focuses on the LoLa ontology, which formally describes DOL's vocabulary for logics, ontology languages (and their serialisations), as well as logic translations. Interestingly, to adequately formalise the logical relationships between these notions, LoLa itself needs to be axiomatised heterogeneously—a task for which we choose DOL. Namely, we use the logic RDF for ABox assertions, OWL for basic axiomatisations of various modules concerning logics, languages, and translations, FOL for capturing certain closure rules that are not expressible in OWL[4], and circumscription for minimising the extension of concepts describing default translations.

1 The Distributed Ontology Language (DOL) – Overview

An ontology in the Distributed Ontology Language (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.[5] Such links include (heterogeneous) *imports* and *alignments*, *conservative extensions* (important for the study of 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.

DOL is standardised within the OntoIOp (Ontology Integration and Interoperability) activity of ISO/TC 37/SC 3[6]. The international working group comprises around 50 experts (around 15 active contributors so far), representing

[4] For the sake of tool availability it is still helpful not to map everything to FOL.
[5] The languages that we call "basic" ontology languages here are usually limited to one logic and do not provide meta-theoretical constructs.
[6] TC = technical committee, SC = subcommittee

a large number of communities in ontological research and application, such as different (1) ontology languages and logics (e.g. Common Logic and OWL), (2) conceptual and theoretical foundations (e.g. model theory, proof theory), (3) technical foundations (e.g. ontology engineering methodologies and linked open data), and (4) application areas (e.g. manufacturing, bio-medicine, etc.). For details and earlier publications, see the project page at `http://ontoiop.org`.

The OntoIOp/DOL standard is currently in its final working draft stage and will be submitted as a committee draft (the first formal standardisation stage) in September 2012.[7] The final international standard ISO 17347 is scheduled for 2015. The standard specifies syntax, semantics, and conformance criteria:

Syntax: abstract syntax of distributed ontologies and their parts; three concrete syntaxes: a text-oriented one for humans, XML and RDF for exchange among tools and services, where RDF particularly addresses exchange on the Web. Here, we will use the DOL text syntax in listings but also explain the RDF vocabulary; for further details on the DOL syntaxes, see [6].

Semantics: (1) a *direct set-theoretical semantics* for the core of the language, extended by an *institutional and category-theoretic semantics* for advanced features such as ontology combinations (technically co-limits), where basic ontologies keep their original semantics; (2) a *translational semantics*, employing the semantics of the expressive Common Logic ontology language for all basic ontologies, taking advantage of the fact that for all basic ontology languages known so far translations to Common Logic have been specified or are known to exist[8]; (3) finally, there is the option of providing a *collapsed semantics*, where the semantics of the meta-theoretical language level provided by DOL (logically heterogeneous ontologies and links between them) is not just specified in semiformal mathematical textbook style, but once more formalised in Common Logic, thus in principle allowing for machine verification of meta properties. For details about the semantics, see [9].

Conformance criteria provide for DOL's extensibility to other basic ontology languages than those considered so far, including future ones. (1) A *basic ontology language* conforms with DOL if its underlying logic has a set-theoretic or, for the advanced features, an institutional semantics. Similar criteria apply to translations between languages. (2) A concrete syntax (*serialisation*) of a basic ontology language conforms if it supports IRIs (Unicode-aware Web-scalable identifiers) for symbols and satisfies further well-formedness criteria. (3) A *document* conforms if it is well-formed w.r.t. one of the DOL concrete syntaxes, which particularly requires explicitly mentioning all logics and translations employed. (4) An *application* essentially conforms if it is capable of processing conforming documents, and providing logical information that is implied by the formal semantics.

[7] The standard draft itself is not publicly available, but ISO/TC 37 has passed a resolution to make the final standard document open, as has been done with the related Common Logic standard [3].

[8] Even for higher-order logics this works, in principle, by using combinators.

2 A Graph of Logic Translations

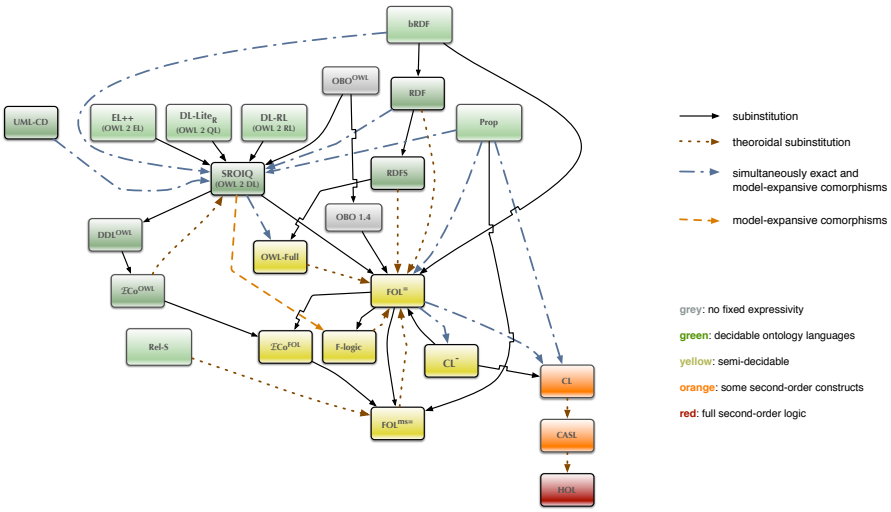

Fig. 1. The logic translation graph for the DOL-conforming languages

Fig. 1 is a revised and extended version of the graph of logics and translations introduced in [8]. 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 all of these logics in earlier publications [9, 8]. The definitions of the DOL-conformance of some central standard ontology languages and translations among them will be given as annexes to the standard, whereas the majority will be maintained in an open registry (cf. Sec. 3). Sec. 4 provides a more fine-grained view on translations (and projections).

3 A Registry for Ontology Languages and Mappings

The OntoIOp standard is not limited to a fixed set of ontology languages. It will be possible to use any (future) ontology language, logic, serialisation, or mapping (translation or projection) with DOL, once its conformance with the criteria specified in the standard has been established. This led to the idea of setting up a *registry* to which the community can contribute descriptions of any such languages, logics, serialisations, or mappings. In the current, early phase of the standardisation process, *we* are maintaining this registry manually. With the release of the final international standard, everyone will be able to make contributions, which an editorial board will review and approve or reject. Fig. 2

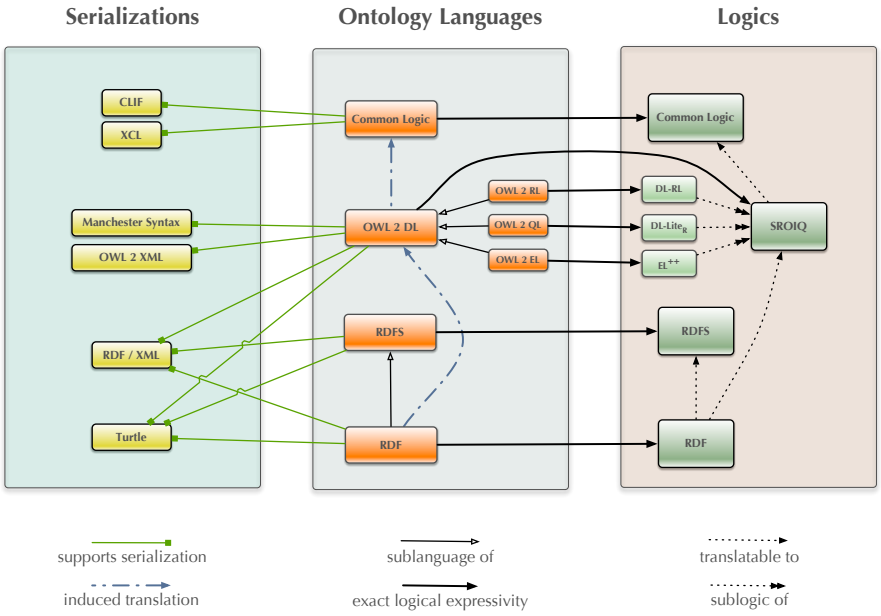

Fig. 2. Subset of the OntoIOp registry, shown as an RDF graph

shows a subset of the OntoIOp registry: a subgraph of Fig. 1 in the "logic" column, as well as related ontology languages and serialisations. Note that the relation between ontology languages and logics generally is *not* bijective: e.g. first-order logic is supported by various languages such as Common Logic, TPTP and CASL.

Any entry of the registry shall be identified by an IRI, so that DOL ontologies can refer to it. At these IRIs, when treated as URLs, there shall be a machine-readable description of the resource according to the linked data principles (cf. [5]), so that, e.g., any agent given a basic ontology can find out the languages this ontology can be translated into (cf. Sec. 6 for an example), or that for any language translation, its definition as well as implementations are available. The most widely supported format for linked data is RDF; we have realised the RDF vocabulary for the OntoIOp registry as a subset of the vocabulary used for serialising DOL ontologies as RDF.[9]

Starting with a plain RDFS vocabulary, we soon realised that we could deliver added value to tools supporting DOL by encoding additional information about the semantics of, e.g., translations into the vocabulary using some OWL constructs, and eventually arrived at a richer formalisation that goes beyond

[9] RDF only allows for *describing*, not for fully formally *defining* logics and translations. To that end, we are planning to alternatively offer full formalisations in the richer OMDoc language from the same IRIs.

OWL: the LoLa ontology. To realise the benefit of a machine-comprehensible representation of this semantics in a rich ontology language, consider DOL's understanding of an ontology language translation: Unless a direct translation on the language level has been specified (e.g. from Common Logic to CASL), one can translate an ontology from a language La to La' if the expressivity of these languages is exactly captured by two logics Lo and Lo', and Lo can (possibly transitively) be translated to Lo'. In a plain RDF graph, this would require multiple lookups.

4 Architecture of the LoLa Ontology

LoLa, the ontology of logics and languages, is implemented as a heterogeneous ontology in DOL, consisting of the following modules:

- An **OWL** core provides classes and properties for the basic concepts, including a basic axiomatisation.
- We use additional **FOL** axioms for closure rules not expressible in OWL, such as non-expressible role compositions.
- We use **circumscription** [7, 1] for minimising the extension of default translations.

The OntoIOp registry, is implemented as an RDF dataset, acting as the ABox of the LoLa ontology. The OntoIOp registry is available through a collection of linked data IRIs in the paths `http://purl.net/dol/{logics,languages,serializations,translations}`, e.g. `http://purl.net/dol/logics/SROIQ` for the logic $\mathcal{SROIQ}$. We made it originally available in RDF/XML, the most widely supported linked data format, but other syntaxes can be provided as well. It can be browsed with frontends like uriburner; try, e.g., `http://linkeddata.uriburner.com/about/html/http/purl.net/dol/logics/SROIQ`.

The OWL core of the LoLa ontology comprises classes for ontology languages, logics, mappings (translations or projections) between ontology languages and between logics, as well as serialisations. The LoLa properties relate all of the former classes to each other, as shown in Fig. 2, e.g. an ontology language to the serialisations that it supports, or to the logic that exactly formalises its expressivity, or an ontology language mapping to the logic mapping it has been derived from.

Fig. 3 shows the top-level classes of LoLa's OWL module, axiomatising logics, languages, and mappings to the extent possible in OWL. Concerning meta-level classes (that is, classes for describing the graph of languages and logics), Fig. 2 already has illustrated the interplay of ontology languages, logics and serialisations.

Object-level classes (that is, classes providing the vocabulary for expressing distributed ontologies) comprise ontologies, their constituents (namely entities, such as classes and object properties, and sentences, such as class subsumptions), as well as links between ontologies.

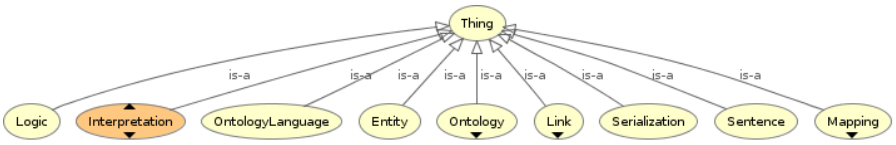

Fig. 3. Top-level classes in the OWL ontology

Mappings are modelled by a hierarchy of properties corresponding to the different types of edges in Fig. 1. For example, object properties such as `translatableTo` model the existence of a translation between two languages. `mappableToLanguage` models the fact that a language can be mapped to another one.

However, this only allows for covering the default translations between logics. E.g., we can express that the default translation from $\mathcal{SROIQ}$ to F-logic is a model-expansive comorphism. Besides further alternative translations that the community may contribute, there is, however, also another translation, which can be obtained from our graph, by composing the $\mathcal{SROIQ} \rightarrow$FOL$^=$ and FOL$^=\rightarrow$F-logic translations, resulting in a subinstitution. For expressing such alternatives, LoLa additionally reifies mappings into classes, whose hierarchy corresponds to that of the mapping properties.

Fig. 4 shows the inferred class hierarchy below the class `Mapping`, as computed within PROTÉGÉ. Notice that our ontology produces several cases of multiple inheritance. Mappings are split along the following dichotomies:

- *logic mapping* versus *ontology language mapping*, cf. Fig. 2.
- *translation* versus *projection*: a translation embeds or encodes an ontology 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 [4].
- *plain mapping* versus *simple theoroidal mapping* [4]: 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 roles etc.

Moreover, we have a class `DefaultMapping` for mappings that are assumed automatically as default when no mapping is given in a certain context.

Other classes concern the accuracy of the mapping, see [8] for details. These classes have mainly been introduced for the classification of logic mappings; however, via the correspondence between logics (mappings) and ontology languages (mappings), they apply to ontology languages as well. *Sublogics* are the most accurate mappings: they are just syntactic subsets. *Embeddings* come close to sublogics, like injective functions come close to subsets. If the model translation is surjective ("model expansion") or even bijective, the mapping is *faithful* in the sense that logical consequence is preserved and reflected, that is, inference

systems and 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 [2].

The full OWL ontology is available at `http://purl.net/dol/1.0/rdf#`; it serves, as said above, simultaneously as an RDF vocabulary for the linked dataset that constitutes the OntoIOp registry, and for serialising DOL ontologies in RDF—therefore the "rdf" name.

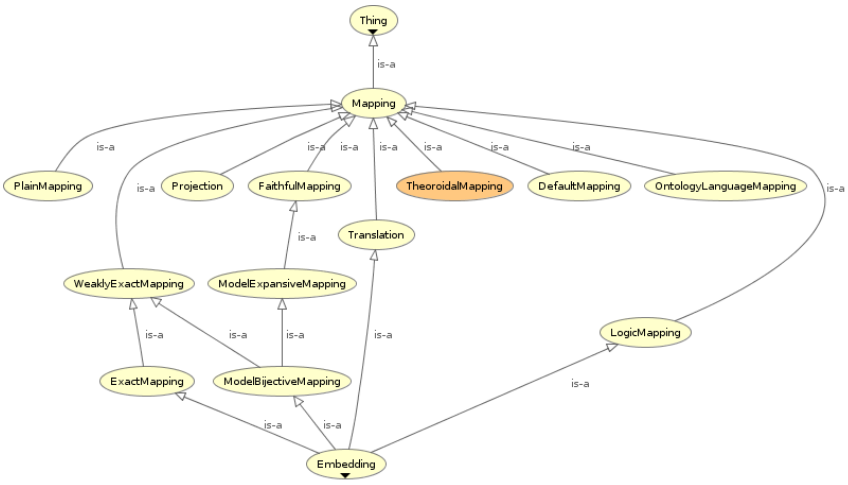

Fig. 4. The part of the OWL ontology concerning mappings

5 Putting It Together in DOL

DOL allows us to put together the pieces collected so far. First, we specify that the RDF registry conforms with the OWL ontology. This is achieved by projecting the registry from RDF to OWL[10], and stating an interpretation of theories of this into the OWL ontology.

We use circumscription [7, 1] for minimising the extent of the class `Default-Translation` and thus implementing a closed world assumption. This feature has been integrated into DOL in a logic independent way: in OWL, it has the effect that classes and object properties are minimised, while in first-order logic, extensions of predicates are.

[10] Basically, this projection turns the RDF graph into an OWL ABox. Impredicativity is escaped from by splitting names that are used in several roles into several distinct names.

Furthermore, we use first-order logic to formulate logical axioms that exceed the expressiveness of OWL. We here use the Common Logic Interchange Format (CLIF) [3]. One such axiom states that supported logics propagate along language translatability; see the ontology `LoLaRules` below.

```
%prefix( :      <http://purl.net/dol/>
         dol:   <http://purl.net/dol/1.0/rdf#>
         log:   <http://purl.net/dol/logics/>
         ser:   <http://purl.net/dol/serializations/>
         trans: <http://purl.net/dol/translations/> )%

distributed-ontology LoLa

%% projecting the RDF ABox to OWL
ontology ABox = registry hide along RDF2OWL end

%% TBox
ontology TBox = dol: end

%% the RDF registry conforms with the OWL ontology
interpretation conformant : ABox to TBox end

%% integrating RDF ABox with OWL TBox while minimising default mappings
logic log:OWL syntax ser:OWL/Manchester
ontology MinimizedABox =
  ABox and TBox
  minimize DefaultMapping %% circumscription-like minimisation
end

%% first-order rules for infering new facts in the registry
logic log:CommonLogic syntax ser:CommonLogic/CLIF
ontology LoLaRules =
  (forall (subLa superLa lo)
    (if (and (dol:translatableTo subLa superLa)
             (dol:mappableToLanguage subLa superLa)
             (dol:supportsLogic subLa lo))
        (dol:supportsLogic superLa lo)))
  ...
end

%% combining OWL ontology with first-order rules
logic log:CommonLogic syntax ser:CommonLogic/CLIF
ontology LoLa =
  dol: translate with OWL2CommonLogic
and
  LoLaRules
end
```

6 Using LoLa to Query the OntoIOp Registry

DOL-conforming applications can explore and query the OntoIOp registry to find out useful information about the logics and languages of concrete given ontologies, or about logics and languages in general.

The following query in the SPARQL RDF query language, e.g., returns all languages a given ontology is translatable to:

```
PREFIX dol: <http://purl.net/dol/1.0/rdf#>
SELECT DISTINCT ?target-language WHERE {
  # first determine, by querying the ontology itself, its language
  <http://ontohub.org/ontologies/my-ontology>
    dol:language ?theLanguageOfTheGivenOntology .
  # find out everything the language is translatable to
  ?theLanguageOfTheGivenOntology
    dol:translatableTo ?targetLanguage ;
    # just to be sure: We are only interested in mappings to languages.
    dol:mappableToLanguage ?targetLanguage .
  # (The use of two properties is owed to the orthogonal design of LoLa.)
}
```

This query assumes that both the information about the ontology and about the OntoIOp registry are available in RDF and ready to be queried as SPARQL. At the moment this cannot be taken for granted; however, we are working on Ontohub, an ontology repository engine, which we will, at the same time, also use to host the OntoIOp registry instead of the current static file deployment [6].

Aiming at wide tool support, the linked data graph that we deploy has all inferences of the LoLa ontology applied; this means in particular that, from a translation between two logics, it is inferred that the corresponding ontology languages are translatable into each other, and that the transitive closure of the translation graph has been computed. Therefore, the query shown above operates on a plain RDF graph, and the query engine does not have to have further inferencing support built in.

The following query focuses exclusively on the OntoIOp registry. It answers a frequent question in knowledge engineering: Which logic is the right one for formalising my conceptual model? For the sake of this example, we focus on knowledge *representability* and thus assume that a logic is suitable if it has translations from and to many other logics. This ignores questions of availability of reasoners for the respective logics, of tools performing the translations, and of their performance. Such information is not yet available in the OntoIOp registry itself, but could be compiled in a separate linked dataset that the registry would link to.

```
PREFIX dol: <http://purl.net/dol/1.0/rdf#>
SELECT ?logic, COUNT(?targetLogic) AS ?t, COUNT(?sourceLogic) AS ?s WHERE {
  ?logic a dol:Logic ;
         dol:translatableTo ?targetLogic ;
         dol:translatableFrom ?sourceLogic .
} ORDER BY ?t, ?s
```

7 Conclusion and Future Work

We have presented LoLa, an ontology of logics, languages, and mappings between them. This ontology formalises the semantics not only of these aspects of the Distributed Ontology Language DOL, but also of the vocabulary employed in the OntoIOp registry for extending the DOL framework with further logics, languages and mappings. LoLa is a heterogeneous ontology consisting of a core OWL module, which declares the vocabulary and provides a basic formalisation, a Common Logic module providing additional first-order rules; furthermore we employ DOL's logic-independent circumscription facility to minimise the extension of default translations. Along with our plans to publish not only machine-comprehensible descriptions of logics and mappings, but full formalisations, we will also expand LoLa to formalise further features of the DOL language, such as the vocabulary that describes the accuracy of a mapping (cf. Sec. 4).

Acknowledgements

The development of DOL is supported by the German Research Foundation (DFG), Project I1-[OntoSpace] of the SFB/TR 8 "Spatial Cognition"; the first author is additionally supported by EPSRC grant EP/J007498/1. The authors would like to thank the OntoIOp working group within ISO/TC 37/SC 3 for their input, particularly Michael Grüninger for his advice regarding the hierarchy of types of ontology translations.

References

1. Bonatti, Piero A., Carsten Lutz, and Frank Wolter, 'The complexity of circumscription in dls', *J. Artif. Intell. Res. (JAIR)*, 35 (2009), 717–773.
2. Borzyszkowski, Tomasz, 'Logical systems for structured specifications', *Theoretical Computer Science*, 286 (2002), 197–245.
3. 'Common Logic (CL): a framework for a family of logic-based languages', Tech. Rep. 24707, ISO/IEC, 2007.
4. Goguen, Joseph, and Grigore Roşu, 'Institution morphisms', *Formal aspects of computing*, 13 (2002), 274–307.
5. Heath, Tom, and Christian Bizer, *Linked Data: Evolving the Web into a Global Data Space*, 1 edn., Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool, San Rafael, CA, 2011.
6. Lange, Christoph, Till Mossakowski, Oliver Kutz, Christian Galinski, Michael Grüninger, and Daniel Couto Vale, 'The Distributed Ontology Language (DOL): Use cases, syntax, and extensibility', in *Terminology and Knowledge Engineering Conference (TKE)*, 2012.
7. McCarthy, John, 'Circumscription - a form of non-monotonic reasoning', *Artif. Intell.*, 13 (1980), 1–2, 27–39.
8. Mossakowski, Till, and Oliver Kutz, 'The Onto-Logical Translation Graph', in Oliver Kutz, and Thomas Schneider, (eds.), *Modular Ontologies*, IOS, 2011.
9. Mossakowski, Till, Oliver Kutz, and Christoph Lange, 'Three semantics for the core of the distributed ontology language', in *FOIS*, 2012.

Proposal of New Approach for Ontology Modularization

Amir Souissi[1], Walid Chainbi[2] and Khaled Ghedira[3]

[1] Ecole Nationale des Sciences de l'Informatique /SOIE - Manouba - Tunisia
Amir.souissi@planet.tn
[2] Sousse National School of Engineers /SOIE, Sousse 4054 - Tunisia
Walid.chainbi@gmail.com
[3] Institut supérieur de gestion de Tunis /SOIE - Tunisia
Khaled.ghedira@isg.rnu.tn

Ontologies have established themselves as a powerful tool to enable knowledge sharing, and a growing number of applications have benefited from the use of ontologies as a means to achieve semantic interoperability among heterogeneous, distributed systems [1]. With the evolution of cooperative and distributed systems, and the emergence of the semantic Web, ontologies have become an indispensable resource. The number of ontologies available on the Web has also increased due to the appearance of several tools that assist users in creating their ontologies. This has posed problems of understanding and reuse of those resources already difficult to design. A solution was then proposed by the knowledge engineers namely *modularization*. Ontology modularization is crucial to support knowledge reuse on the ever increasing semantic Web [2]. However, modularization methods that serve the reuse goal are often intended for humans to assist them in building new ontologies, rather than for applications that need only a relevant part of an existing ontology. Moreover, modules obtained are always subject to verification and maintenance by humans to validate the semantic consistency of their contents. Unlike previous studies, we investigate in this paper how a modularization based on semantic comparison, may provide a module directly reusable by the application that requests it. Our contribution is twofold. On the one hand, it allows an application to extract and use a module that covers a subdomain from an ontology that covers a wider knowledge area, regardless of its structure and the formalism with which it is expressed. On the other hand, the user is relieved from manually estimating the meaning of the components of the ontology, after the modularization process.

The modularization approach we propose is part of the decomposition approaches of monolithic ontologies [3,4]. It is an *extraction method* since it aims to extract a relevant ontology module. The method should allow the user to express its needs by entering the concepts which interest him. The result is a fragment composed of concepts and relations that are relevant to the module i.e., which are *in strong semantic relationship* with the concepts submitted by the user. We define a *strong semantic relationship* between two concepts, as one of the six logic functions as follows:

- *Identity relationship*: it is a semantic relation between two concepts that have the same syntax, the same attributes and operations. Example: Identity (Person, Person).

- *Synonymy relationship*: it is a semantic relation between two concepts that express the same meaning. Example: Synonymy (Person, Individual).

- *Classification Is-a relationship*: two concepts where one is expressing a particular case of the other. Example: Is-a (Student, Person).

- *Homonymy relationship*: the same concept can have two different meanings. Example: Homonymy (Bug, Bug). The first one means an insect. The second one means a fault in a computer system.

- *Equivalence relationship*: a semantic relationship between two concepts that play the same role. Example: Equivalence (Teacher, Professor).

- *Antonymy relationship*: is used between two concepts totally disjoint. Example: Antonymy (Registered, Visitor).

For example, in an ontology that describes the human anatomy, the user is only interested in the anatomy of the foot. The method should extract a coherent module, semantically rich on the foot, from the ontology of departure.

Our approach is based on two basic steps:

- *1st step*: Identifying concepts that are in strong semantic relationship with external concepts.

- *2nd step*: composition of the module based on the concepts identified in Step 1. All concepts that appear in the definition of the concepts identified are considered part of the module.

The goal is to allow a program to extract automatically a single part of an ontology without human intervention and without restrictions on the ontology structure. This will help programs to satisfy their requirements by reusing directly ontology portions.

References

1. Chainbi, W.: An Ontology Based Multi-Agent System Conceptual Model, In Special Issue of the International Journal of Computer Applications in Technology, Inderscience Publishers, Vol. 31, Nos. 1/2, pp. 35-44, (2008)
2. Cuenca Grau,B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the Right Amount: Extracting Modules from Ontologies. In International World Wide Web conference, (2007)
3. Seidenberg, J.: Web Ontology Segmentation: Extraction, Transformation, Evaluation. In Heiner Stuckenschmidt, Christine Parent, Stefano Spaccapietra (Eds.), Modular Ontologies concepts, Theories and Techniques for Knowledge Modularization, Springer-Verlag Berlin, Heidelberg, LNCS 5445, pages 211-243, (2009)
4. Stuckenschmidt, H., Klein, M.: Strutured-based Partitioning of large concept hierarchies. In Proceedings of the 3rd International Semantic Web Conference, (2004)

Distributed Reasoning with $E^{DDL}_{HQ^+}$ $\mathcal{SHIQ}$.

George A. Vouros[1] and George M.Santipantakis[2]

[1] Digital Systems, University of Piraeus, Greece georgev@unipi.gr
[2] ICSD, University of the Aegean, Greece gsant@aegean.gr

To deal with autonomous agents' knowledge and subjective beliefs in open, heterogeneous and inherently distributed settings, we need special formalisms that combine knowledge from multiple and potentially heterogeneous interconnected contexts. Each context contains a chunk of knowledge defining a logical theory, called *ontology unit*). While standard logics may be used, subjectiveness and heterogeneity issues have been tackled by knowledge representation formalisms called *contextual logics* or *modular ontology languages* (e.g. [1] [2]).Nevertheless, in distributed and open settings we may expect that different ontology units should be combined in many different, subtle ways without making any assumption about the disjointness of the domains covered by different units. To address this issue we need to increase the expressivity of the language used for defining correspondences.Towards this goal, we have been motivated to propose the representation framework $E^{DDL}_{HQ^+}$ $\mathcal{SHIQ}$ (or simply $E-\mathcal{SHIQ}$).

The $E^{DDL}_{HQ^+}$ $\mathcal{SHIQ}$ framework. Given a finite index set of units' identifiers I, each unit M_i consists of a TBox $\mathcal{T}_i$, RBox $\mathcal{R}_i$, and ABox $\mathcal{A}_i$ in the $\mathcal{SHIQ}$ fragment of Description Logics[3].

Given an $i \in$ I, let N_{C_i}, N_{R_i} and N_{O_i} be the sets of concept, role and individual names respectively. For some $R \in N_{R_i}$, $Inv(R)$ denotes the inverse role of R and $(\mathcal{N}_\mathcal{R} \cup \{Inv(R)|R \in \mathcal{N}_\mathcal{R}\})$ is the set of $\mathcal{SHIQ}$-roles. The set of $\mathcal{SHIQ}$-concepts is the smallest set constructed by the constructors in $\mathcal{SHIQ}$. Cardinality restrictions can be applied on R, given that R is a *simple role*. An interpretation $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}_i, \cdot^{\mathcal{I}_i} \rangle$ consists of a domain $\Delta^{\mathcal{I}_i}_i \neq \emptyset$ and the interpretation function $\cdot^{\mathcal{I}_i}$ which maps every $C \in N_{C_i}$ to $C^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i}_i$, every $R \in N_{R_i}$ to $R^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i}_i \times \Delta^{\mathcal{I}_i}_i$ and each $a \in N_{O_i}$ to an element $a^{\mathcal{I}_i} \in \Delta^{\mathcal{I}_i}_i$. Elements and axioms in unit M_i are denoted by $i : c$. Each Tbox $\mathcal{T}_i$ contains generalized concept inclusion axioms, RBox $\mathcal{R}_i$ contains role inclusion axioms, and ABox $\mathcal{A}_i$ contains assertions for individuals and their relations [3].

Towards combining knowledge in different units, the proposed framework allows the connection of units via: (a) concept-to-concept subjective correspondences [1] specified by *onto*-bridge rules $i : C \xrightarrow{\sqsupseteq} j : G$, or *into*-bridge rules $i : C \xrightarrow{\sqsubseteq} j : G$, where $i \neq j \in$ I. (b) Individual subjective correspondences $i : a_i \xrightarrow{=} j : b_j$, where $a_i \in N_{O_i}$ and $b_j \in N_{O_j}$. The above mentioned subjective correspondences concern the point of view of M_j. (c) Link-properties [2](or *ij-properties*, $i, j \in$ I), which can be related via ij-property inclusion axioms, be transitive and, if they are simple, be restricted by qualitative restrictions. The sets of ij-properties' names, i.e. the sets ϵ_{ij}, $i, j \in$ I, are not necessarily pair-

wise disjoint, but disjoint with respect to N_{C_i}, and N_{O_i}. A set of ij-properties connecting concepts of M_i with concepts of M_j, is defined as the set $\mathcal{E}_{ij} = \epsilon_{ij}$, $i \neq j \in I$, and in case $i = j$, it is the set $\mathcal{E}_{ij} = \epsilon_{ij} \cup \{Inv(E)|E \in \epsilon_{ji}\}$, where ϵ_{ij} is the set of (local to M_i) role names. ij-properties are being used for specifying concepts (so called $i-$concepts) in the M_i unit.

Transitive axioms are of the form $Trans(E; (i, j))$, where $E \in \mathcal{E}_{ij} \cap \mathcal{E}_{ii}$, E is transitive in M_i and transitive ij-property. Transitivity axioms and the finite set of inclusion axioms for ij-properties form the ij-property box $\mathcal{R}_{ij}$ (if $i = j$, $\mathcal{R}_{ii} = \mathcal{R}_i$). The combined property box RBox $\mathcal{R}$ is a family of ij-property boxes. A *combined TBox* is a family of TBoxes $\mathbf{T}= \{\mathcal{T}_i\}_{i\in I}$. A *distributed ABox* $\mathbf{A} = \{\mathcal{A}_i\}_{i\in I}$, includes a collection of individual correspondences, and property assertions of the form $(a \cdot E_{ij} \cdot b)$, where $E_{ij} \in \mathcal{E}_{ij}$. A *distributed knowledge base* Σ is composed as $\Sigma = \langle \mathbf{T}, \mathcal{R}, \mathfrak{B}, \mathbf{A}\rangle$, where $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{i\neq j\in I}$ is the collection of bridge rules between ontology units. Each $\mathcal{R}_{ij}$, is interpreted by a valuation function $\cdot^{\mathcal{I}_{ij}}$ that maps every ij-property to a subset of $\Delta_i^{\mathcal{I}_i} \times \Delta_j^{\mathcal{I}_j}$. Let $\mathcal{I}_{ij} = \langle \Delta_i^{\mathcal{I}_i}, \Delta_j^{\mathcal{I}_j}, \cdot^{\mathcal{I}_{ij}} \rangle$, $i, j \in I$. It must be noted that, for a specific $i \in I$ and a property E in the i-th unit, this property may be shared between different ij-property boxes (i.e. for different j's). In this case, the denotation of E is $\bigcup_{j\in I} E^{\mathcal{I}_{ij}}$. A *domain relation* $r_{ij}, i \neq j$ from $\Delta_i^{\mathcal{I}_i}$ to $\Delta_j^{\mathcal{I}_j}$ is a subset of $\Delta_i^{\mathcal{I}_i} \times \Delta_j^{\mathcal{I}_j}$, s.t. for each $d \in \Delta_i^{\mathcal{I}_i}$, $r_{ij}(d) \subseteq \{d'|d' \in \Delta_j^{\mathcal{I}_j}\}$, and in case $d' \in r_{ij}(d_1)$ and $d' \in r_{ij}(d_2)$, then $d_1 = d_2$. For a subset D of $\Delta_i^{\mathcal{I}_i}$, $r_{ij}(D)$ denotes $\cup_{d\in D} r_{ij}(d)$. A domain relation represents only equalities, i.e. each $d_1 \in r_{ij}(d)$ is equal to the other individuals in $r_{ij}(d)$. The distributed knowledge base is interpreted by a *Distributed Interpretation*, $\mathfrak{I}$ s.t. $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i\in I}, \{\mathcal{I}_{ij}\}_{i,j\in I}, \{r_{ij}\}_{i\neq j\in I}\rangle$.

We have specified a sound and complete distributed Tableau algorithm that has been implemented by extending the Pellet reasoner [3]. The instance retrieval algorithm for the framework has been presented in [4].

Acknowledgement: This research project 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. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. Journal of Data Semantics **1** (2003) 153–184
2. Parsia, B., Cuenca Grau, B.: Generalized link properties for expressive epsilon-connections of description logics. In: AAAI. (2005) 657–662
3. Baader, F.e.a., ed.: The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press (2003)
4. Santipantakis, G., Vouros, G.: Distributed Instance Retrieval in $E_{HQ^+}^{DDL}\mathcal{SHIQ}$ Representation Framework. In: Artificial Intelligence: Theories and Applications. Volume 7297 of LNCS. Springer Berlin / Heidelberg (2012) 141–148

[3] The full paper describing the framework and the tableau algorithm can be found in http://ai-lab-webserver.aegean.gr/gsant/ESHIQ.Report