DATAK 982 ARTICLE IN PRESS No. of Pages 24, Model 3+
7 March 2007 Disk Used

Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Data & Knowledge Engineering xxx (2007) xxx–xxx
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Reasoning and change management in modular ontologies

Heiner Stuckenschmidt [a,*], Michel Klein [b]

[a] Universität Mannheim, Germany
[b] Vrije Universiteit Amsterdam, The Netherlands

Received 13 December 2006; accepted 5 February 2007

Abstract

The benefits of modular representations are well known from many areas of computer science. While in software engineering modularization is mainly a vehicle for supporting distributed development and re-use, in knowledge representation, the main goal of modularization is efficiency of reasoning. In this paper, we concentrate on the benefits of modularization in the context of ontologies, explicit representations of the terminology used in a domain. We define a formal representation for modular ontologies based on the notion of Distributed Description Logics and introduce an architecture that supports local reasoning by compiling implied axioms. We further address the problem of guaranteeing the correctness and completeness of compiled knowledge in the presence of changes in different modules. We propose a heuristic for analyzing changes and their impact on compiled knowledge and guiding the process of updating compiled information that can often reduce the effort of maintaining a modular ontology by avoiding unnecessary re-compilation.

Keywords: Ontologies; Reasoning; Distributed Knowledge Representation; Change management

1. Motivation

Currently, research in the area of the semantic web is in a state where ontologies are ready to be applied in real applications such as semantic web portals, information retrieval or information integration. In order to lower the effort of building ontology-based applications, there is a clear need for a representational and computational infrastructure in terms of general purpose tools for building, storing and accessing ontologies. A number of such tools have been developed, i.e. ontology editors [1,2], reasoning systems [3,4] and more recently storage and query systems (e.g. [5]). Most of these tools, however, treat ontologies as mono-

* Corresponding author. Address: Universität, Mannheim, Institut für Informatik, A5, 6, 68159 Mannheim, DE, Germany. Tel.: +49 621 181 2530.
E-mail address: heiner@informatik.uni-mannheim.de (H. Stuckenschmidt).

0169-023X/$ - see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.datak.2007.02.001

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

lithic entities and provide little support for specifying, storing and accessing ontologies in a modular manner.

1.1. Why modularization?

There are many reasons for thinking about ontology modularization. Our work is mainly driven by three arguments. These also bias the solution we propose, as it focuses on the following aspects.

Distributed Systems: In distributed environments like the semantic web, the question for modularization arises naturally. Ontologies in different places are built independent of each other and can be assumed to be highly heterogeneous. Unrestricted referencing of concepts in a remote ontology can therefore lead to serious semantic problems as the domain of interpretation may differ even if concepts appear to be the same on a conceptual level. The introduction of modules with local semantics can help to overcome this problem.

Large Ontologies: Modularization is not only desirable in distributed environments, it also helps to manage very large ontologies that we find in medicine or biology. These ontologies, which sometimes contain more than a hundred thousand concepts, are hard to maintain as changes are not contained locally but can affect large parts of the model. Another argument for modularization in the presence of large ontologies is re-use: in most cases, we are not interested in the complete ontology when building a new system, but only in a specific part. Experiences from software engineering shows that modules provide a good level of abstraction to support maintenance and re-use.

Efficient reasoning: A specific problem with distributed ontologies as well as with very large models is the efficiency of reasoning. While the pure size of the ontologies causes problems in the latter case, hidden dependencies and cyclic references can cause serious problems in a distributed setting. The introduction of modules with local semantics and clear interfaces will help to analyze distributed systems and provides a basis for the development of methods for localizing inference.

1.2. Requirements

There are three requirements a modular ontology architecture has to fulfill in order to improve ontology maintenance and reasoning in the way suggested above. The requirements will be the main guidelines for the design of our solution proposed in this work.

Loose Coupling: In general, we cannot assume that two ontology modules have anything in common. This holds for the conceptualization as well as for the interpretation of objects, concepts or relations. Our architecture has to reflect this by providing an extremely loose coupling of modules. In particular, we have to prevent unwanted interactions between modules. For this purpose, mappings between modules have to be distinguished from local definitions on the semantic as well as the conceptual level.

Self-Containment: In order to facilitate the re-use of individual modules from a larger, possibly interconnected system, we have to make sure that modules are self-contained. In particular, it should be possible to perform certain reasoning tasks such as subsumption or query answering within a single module without having to access other modules. This is also important if we want to provide efficient reasoning. Further, we have to ensure correctness, and whenever possible completeness, of local reasoning, for obvious reasons.

Integrity: The advantages of having self-contained ontology modules have their price in terms of potential inconsistencies that arise from changes in other ontology modules. While there is in our architecture no need to access other modules at reasoning time, the correctness of reasoning within a self contained module may still depend on knowledge in other ontologies. If this knowledge changes, reasoning results in a self-contained module may become incorrect with respect to the overall system, and we will not even notice it.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

We have to provide mechanisms for checking whether relevant knowledge in other systems has changed and for adapting the reasoning process if needed to ensure correctness.

1.3. Related work

Our work relates to two main areas of research on representing and reasoning about ontological knowledge. The first is concerned with distributed and modular knowledge representation where we use ideas from theorem proving and knowledge engineering. The second area of related work is concerned with managing knowledge models. Here previous work exists in the area of knowledge engineering and semantic web technologies.

While the principle of modularity has widely been adopted in software engineering it has got less attention in the area of knowledge representation and reasoning. Some fundamental work on the modularization of representations can be found in the area of theorem proving. Farmer and colleagues promote the use of combinations of 'Little Theories', representations of a specific mathematical structure in order to reason about complex problems [6]. They show the advantages of this modular approach in terms of reusability and reduced modeling effort. The idea of reusing and combining chunks of knowledge rather than building knowledge bases from scratch has later been adopted by the knowledge engineering community for building real-world knowledge bases (e.g. see [7]). McIlraith and Amir argue that a modularization of knowledge bases has also advantages for reasoning, even if the modularization is done a posteriori. They present algorithms for breaking down existing representations into a set of modules with minimal interaction and define reasoning procedures for propositional [8] and first-order logic [9]. The work reported there is motivated by well established techniques from uncertain reasoning, where an a posteriori modularization of large theories is a common way to reduce runtime complexity (e.g. see [10]).

As we are interested in representations of ontological knowledge, approaches from the area of logics for representing terminologies, so-called description logics are of special interest for our work. In this area, we find the same arguments for a modularized representation as in the area of theorem proving. Rector proposes a strategy for modular implementation of ontologies using description logics [11]. The approach is based on a set of orthogonal taxonomies that provide a basis for defining more complex concepts. Rector argues for the benefits of this strategy in terms of easier creation and re-use of ontological knowledge. Buchheit and others propose a similar structuring on the language level by dividing the terminological part of a knowledge base into a schema part that corresponds to the basic taxonomies and a view part [12]. They show that this distinction can be used to achieve better run-time behavior for complex view languages. While these approaches still assume the overall model to be a single ontology providing a coherent conceptualization of the world, Giunchiglia and others propose a more radical approach to distributed representations. They propose the local model semantics as an extension of the standard semantics of first order logics [13]. This semantics allows different modules to represent different views on the same part of the world and the definition of directed partial mappings between different modules. Recently, Borgida and Serafini defined a distributed version of description logics based on local model semantics that has all advantages of the contextual representations [14].

As already mentioned, the problem of combining and reasoning with ontological modules has become of central importance in research on knowledge representation and reasoning on the semantic web. Standard languages for encoding ontological knowledge on the World Wide Web, i.e. the RDF schema [15] and the web ontology language OWL [16] provide some basic mechanisms for combining modular representations. The abilities to combine different models are restricted to the import of complete models and to the use of elements from a different model in definitions by direct reference. It is assumed that references to external statements are only made for statements from imported models, however, this is strictly speaking not required. As a consequence, mappings rather implicitly exist in terms of mutual use of statements across models. In [17] the authors provide a detailed analysis of the drawbacks of using OWL import statements for combining heterogeneous ontologies and show that local model semantics, which is also the basis for our work solves these problems. Volz and colleagues discuss different interpretations of the import statement that range from purely syntactic to schema-aware interpretations of the imported knowledge [18]. An alternative way of relating dif-

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

ferent RDF models to each others that is much closer to our ideas is discussed by Oberle [19] who defines a view language for RDF.

Recently, there has been some interest in formal models for modular ontologies that can be seen as competing approaches to the one described in this paper. In particular, researchers have proposed the use of ϵ-connections [20] as a suitable formalism for inter-module links and investigated the logical properties of ontologies. Others have proposed to strengthen distributed description logics by adding requirements to the links between different local models resulting in a formalism called P-DL [21]. These models, however, take a slightly different view on modular ontologies. While in this paper, we assume that modular ontologies are created by linking previously unrelated, possibly inconsistent ontologies that can also have overlap in their scope, the above mentioned approaches focus on a scenario where an existing ontology is partitioned into a number of modules for the sake of enhancing (re-)usability and thus assume a tighter coupling of the different modules which requires a stronger formalism for specifying links between the modules.

1.4. Our approach

In the following, we describe our approach to ontology modularization on an abstract level. We emphasize the main design decisions and motivate them on the basis of the requirements defined above. The technical details of the approach will be given in the subsequent sections.

- *View-Based Mappings*: The first design decision concerns the way different ontology modules are connected. In our work, we adopt the approach of view-based information integration. In particular, ontology modules are connected by conjunctive queries and the extension of a concept in one module can be claimed to be equivalent to the (intentional) answer set of a conjunctive query over the vocabulary of another module. This way of connecting modules is more expressive than simple one-to-one mappings between concept names. Further, the same technique can be used to define relations of any arity based on other modules. Compared to the use of arbitrary axioms, our approach is less expressive. We decide to sacrifice a higher expressiveness for the sake of conceptual simplicity and desirable semantic properties that are discussed in the remainder of this paper.
- *Interface Compilation*: The use of conjunctive queries guarantees a loose coupling on a conceptual and semantic level. However, it does not provide self-containment, because reasoning in an ontology module depends on the answer sets of the queries that are used to connect it to other modules. These answer sets have to be determined by actually querying the other ontology module. In order to make local reasoning independent from other modules, we use a knowledge compilation approach. The idea is to compute the result of each mapping query off-line and add the result as an axiom to the ontology module. At reasoning time these axioms replace the query, thus enabling local reasoning. As the results of queries are considered to be defined intentionally rather than extensionally, the result of the compilation of a query is not a set of instances retrieved from other modules, but a set of axioms that contains all the information necessary to perform local reasoning.
- *Change Detection and Automatic Update*: Our approach of compiling mappings and adding the result to the different ontology modules is very sensitive against changes in ontology modules. Once a query has been compiled, the correctness of reasoning can only be guaranteed as long as the queried ontology module does not change. On the other hand, not every change in the system does really influence the compiled result. Problems only arise if concepts used in the query change or if the set of concepts subsuming the query is changed. In the second case, we will have to compile the interface again. In the first case we might even have to consider a redefinition of the query. In order to decide whether the compiled axiom is still valid, we propose a change detection mechanism that is based on a taxonomy of ontological changes and their impact on the concept hierarchy in combination with the position of the affected concept in the hierarchy.

In the following, we first introduce a representational framework for modular ontologies that builds on top of existing work on distributed description logics (DDL) as a framework for reasoning about distributed ontologies. In Section 3 we define reasoning mechanisms for modular ontologies as a special case of general inference in distributed description logics. We further introduce the compilation of implied subsumption rela-

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

Table 1
Axiom patterns for describing ontologies in description logics

DL Axiom	Semantics	Intuition
A-Box		
$C(x)$	$x^{\mathscr{I}} \in C^{\mathscr{I}}$	x is of type C
$P(x,y)$	$(x^{\mathscr{I}}, y^{\mathscr{I}}) \in P^{\mathscr{I}}$	x is related to y by R
T-Box		
$C \sqsubseteq D$	$C^{\mathscr{I}} \subseteq D^{\mathscr{I}}$	C is more specific than D
$p \sqsubseteq r$	$p^{\mathscr{I}} \subseteq r^{\mathscr{I}}$	P is more specific than R
$p \equiv r^{-}$	$p^{\mathscr{I}} = \{(x,y) \mid (y,x) \in r^{\mathscr{I}}\}$	P is the inverse of R

tions as a mechanisms for localizing reasoning and compare it with the distributed reasoning methods proposed for DDL. Section 4 discusses the problem of handling changes in external ontologies and their impact on compiled knowledge and proposes a heuristic for checking whether compiled knowledge has to be recomputed. We conclude with an example from a case study on ontology evolution in Section 5 and a discussion of the tradeoffs of our approach and possible extensions in Section 7.

2. Modular ontologies

In this section, we present a formal model for modular ontologies that will be used throughout the paper. Our starting point is the use of description logics – a special kind of logics for representing terminological knowledge as the basis for representing ontologies. We briefly explain the nature of description logics and their semantics. We then formally introduce the logic $\mathscr{SHIQ}$ which is the basis for our work. We then proceed with the definition of our model for modular ontologies by briefly recalling an extension of $\mathscr{SHIQ}$ with mappings between different models known as Distributed Description Logics (DDL). As our model turns out to be a subset of Distributed Description Logics, we conclude this section by explaining the restrictions to the general framework of DDL that apply to the modular ontologies.

2.1. Ontologies and description logics

An Ontology usually groups objects of the World that have certain properties in common (e.g. cities or countries) into concepts. A specification of the shared properties that are characteristic for this set of objects is called a concept definition. Concepts can be arranged into a subclass–superclass relation in order to be able to further discriminate objects into subgroups (e.g. capitals or European countries). Concepts can be defined in two ways, by enumeration of its members or by a concept expression. The specific logical operators that can be used to formulate concept expressions can vary between ontology languages (Table 1).

Description Logics are decidable subsets of first order logic that are designed to describe concepts in terms of complex logical expressions[1] The basic modeling elements in Description Logics are concepts (classes of objects), roles (binary relations between objects) and individuals (named objects). Based on these modeling elements, Description Logics contain operators for specifying so-called concept expressions that can be used to specify necessary and sufficient conditions for membership in the concept they describe. These modeling elements are provided with a formal semantics in terms of an abstract domain interpretation mapping $\mathscr{I}$ mapping each instance onto an element of an abstract domain $\Delta^{\mathscr{I}}$. Instances can be connected by binary relations defined as subsets of $\Delta^{\mathscr{I}} \times \Delta^{\mathscr{I}}$. Concepts are interpreted as a subset of the abstract domain Δ. Intuitively, a concept is a set of instances that share certain properties. These properties are defined in terms of concept expressions. Typical operators are the Boolean operators as well as universal and existential quantification over relations to instances in other concepts.

A Description Logic Knowledge base consists of two parts. The A-box contains information about objects, their type and relations between them, the so-called T-Box consists of a set of axioms about concepts (potentially defined in terms of complex concept expressions and relations). The first type of axioms can be used to

[1] Details about the relation between description logics and first order logic can be found in [22,23].

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

describe instances. In particular, axioms can be used to state that an instance belongs to a concept or that two instances are in a certain relation. It is easy to see, that these axioms can be used to capture case descriptions as labeled graphs. The other types of axioms describe relations between concepts and instances. It can be stated that one concept is a subconcept of the other (all its instances are also instances of this other concept). Further, we can define a relation to be a subrelation or the inverse of another relation. These axioms are used to formalize the legal ontology described in the last section.

The formal semantics of concepts and relations as defined by the interpretation into the abstract domain $\Delta^{\mathcal{I}}$ can be used to automatically infer new axioms from existing definitions.

It has been argued that encoding ontologies in Description Logics is beneficial, because it enables inference engines to reason about ontological definitions. In this context, deciding subsumption between two concept expressions, i.e. deciding whether one expression is more general than the other one is one of the most important reasoning tasks as it has been used to support various tasks including information integration [24], product and service matching [25,26] and Query answering over ontologies [27].

In this paper, we consider ontologies represented in the description logic $\mathcal{SHIQ}$. This choice is motivated by the fact that $\mathcal{SHIQ}$ covers a large part of the expressive power of the Web Ontology Language OWL [16], more specifically of the language OWL-DL, a decidable sublanguage of OWL that directly corresponds to the logic $\mathcal{SHOIQ}$ that extends $\mathcal{SHIQ}$ with nominals [28]. We omit this extension in order to be able to base our framework on recent results on Distributed Description Logics [29,30] that provide us with basic mechanisms for specifying links between concepts in different ontologies in a loose way. Before defining our notion of modular ontologies, we briefly introduce the logic $\mathcal{SHIQ}$ as well as the basic notions of Distributed Description Logics. For further information about notation and naming in Description Logics, we refer to [31].

2.2. The $\mathcal{SHIQ}$ description logic

Let $\mathcal{C}$ be a set of concept names and RN a set of role names. Further let there be a set $R^+ \subseteq RN$ of transitive roles (i.e. for each $r \in R^+$ we have $r(x,y) \wedge r(y,z) \Rightarrow r(x,z)$). If now r^- denotes the inverse of a role (i.e. $r(x,y) \Rightarrow r^-(y,x)$) then we define the set of roles R as $RN \cup \{r^- \mid r \in RN\}$. A role inclusion axiom is an expression $r \sqsubseteq s$ where r and s are roles. A role is called a simple role if it is not transitive and does not have transitive subroles with respect to the transitive closure of the role inclusion relation. Concept expressions are now formed by applying special operators to concept and role names. In particular, new concept expressions can be formed from existing ones using the Boolean operators or by imposing constraints on the type and number of objects related to objects of the described concept. The corresponding operators are summarized below

Expression	Intuition
$\neg C$	All objects that are not of type C
$C \sqcap D$	All objects that are of type C and of type D
$C \sqcup D$	All Objects that are of type C or of type D
$\exists r.C$	All Objects that related to some objects of type C via relation r
$\forall r.C$	All Objects that are only related to objects of type C via relation r
$(\geqslant nr.C)$	All objects that are related to at least n objects of type C via relation r
$(\leqslant nr.C)$	All objects that are related to at most n objects of type C via relation r

Formally, the set of concepts (or concept expressions) in $\mathcal{SHIQ}$ is the smallest set such that:

- $\top$ and $\bot$ are concept expressions for the most general concept and the unsatisfiable concept, respectively;

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

- every concept name A is a concept expression;
- if C and D are concept expressions, r is a role, s is a simple role and n is a non-negative integer, then $\neg C$, $C \sqcap D$, $C \sqcup D$,$\forall r.C$, $\exists r.C$, $(\geqslant nr.C)$ and $(\leqslant nr.C)$ are concept expressions.

A general concept inclusion axiom is an expression $C \sqsubseteq D$ where C and D are concepts. A terminology is a set of general concept inclusion and role inclusion axioms.

The semantics of $\mathscr{SHIQ}$ is defined in terms of an interpretation $\mathscr{I} = (\Delta^{\mathscr{I}}, .^{\mathscr{I}})$ where $.^{\mathscr{I}}$ is a function that maps every concept on a subset of $\Delta^{\mathscr{I}}$ and every role on a subset of $\Delta^{\mathscr{I}} \times \Delta^{\mathscr{I}}$ such that for all concepts C and D and for roles r where #M denotes the cardinality of M and $(r^{\mathscr{I}})^+$ the transitive closure of $r^{\mathscr{I}}$ we have

- $\top^{\mathscr{I}} = \Delta^{\mathscr{I}}$ and $\bot^{\mathscr{I}} = \emptyset$
- $r^{\mathscr{I}} = (r^{\mathscr{I}})^+$ for $r \in R^+$ and $r^- = \{(y,x)|(x,y) \in r^{\mathscr{I}}\}$
- $(\neg C)^{\mathscr{I}} = \Delta^{\mathscr{I}} - C^{\mathscr{I}}$, $(C \sqcap D) = C^{\mathscr{I}} \cap D^{\mathscr{I}}$ and $(C \sqcup D)^{\mathscr{I}} = C^{\mathscr{I}} \cup D^{\mathscr{I}}$
- $(\forall r.C)^{\mathscr{I}} = \{x \mid \forall y.(x,y) \in r^{\mathscr{I}} \Rightarrow y \in C^{\mathscr{I}}\}$
- $(\exists r.C)^{\mathscr{I}} = \{x \mid \exists y.(x,y) \in r^{\mathscr{I}} \wedge y \in C^{\mathscr{I}}\}$
- $(\geqslant nr.C)^{\mathscr{I}} = \{x \mid \#\{y.(x,y) \in r^{\mathscr{I}} \wedge y \in C^{\mathscr{I}}\} \geqslant n\}$
- $(\leqslant nr.C)^{\mathscr{I}} = \{x \mid \#\{y.(x,y) \in r^{\mathscr{I}} \wedge y \in C^{\mathscr{I}}\} \leqslant n\}$

An interpretation satisfies a terminology $\mathscr{T}$ if $C^{\mathscr{I}} \subseteq D^{\mathscr{I}}$ for all general concept inclusions $C \sqsubseteq D$ in $\mathscr{T}$ and $r^{\mathscr{I}} \subseteq s^{\mathscr{I}}$ for all role inclusion axioms $r \sqsubseteq s$ in $\mathscr{T}$. In this case we call $\mathscr{I}$ a model for $\mathscr{T}$. A concept D subsumes a concept C in $\mathscr{T}$ if $C \sqsubseteq D$ holds for all models of $\mathscr{T}$. In the remainder of the paper we will focus on the task of deciding whether a concept subsumes another one.

2.3. Distributed description logic

Distributed Description Logics as proposed in [29] provide a language for representing sets of terminologies and semantic relations between them. For this purpose DDLs provide mechanisms for referring to terminologies and for defining rules that connect concepts in different terminologies. On the semantic level, DDLs extend the notion of interpretation introduced above to fit the distributed nature of the model and to reason about concept subsumption across terminologies.

Let I be a non-empty set of indices and $\{\mathscr{T}_i\}_{i \in I}$ a set of terminologies. We prefix inclusion axioms with the index of the terminology they belong to (i.e. $i : C$ denotes a concept in terminology $\mathscr{T}_i$ and $j : C \sqsubseteq D$ a concept inclusion axiom from terminology $\mathscr{T}_j$). Note that $i : C$ and $j : C$ are different concepts. Semantic relations between concepts in different terminologies are represented in terms of axioms of the following form, where C and D are concepts in terminologies $\mathscr{T}_i$ and $\mathscr{T}_j$, respectively:

- $i : C \xrightarrow{\sqsubseteq} j : D$ (into-rule)
- $i : C \xrightarrow{\sqsupseteq} j : D$ (onto-rule)

These axioms are also called *bridge-rules*. The into-rule states that concept C in terminology $\mathscr{T}_i$ is intended to be more specific than concept D in terminology $\mathscr{T}_j$. Conversely, the onto-rule states that concept C in terminology $\mathscr{T}_i$ is intended to be more general than concept D in terminology $\mathscr{T}_j$. An additional rule $i : C \xrightarrow{\equiv} j : D$ is defined as the conjunction of the two rules above, stating that the two concepts are intended to be equivalent. A distributed terminology $\mathfrak{T}$ is now defined as a pair $(\{\mathscr{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I})$ where $\{\mathscr{T}_i\}_{i \in I}$ is a set of terminologies and $\{\mathfrak{B}_{ij}\}_{i \neq j \in I}$ is a set of bridge rules between these terminologies.

The semantics of distributed description logics is defined in terms of a global interpretation $\mathfrak{I} = (\{\mathscr{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I})$ where $\mathscr{I}_i$ is an interpretation for terminology $\mathscr{T}_i$ as defined above and $r_{ij} \subseteq \Delta^{\mathscr{I}_i} \times \Delta^{\mathscr{I}_j}$ is a domain relation connecting elements of the interpretation domains of terminologies $\mathscr{T}_i$ and $\mathscr{T}_j$. We use $r_{ij}(x)$ to denote $\{y \in \Delta^{\mathscr{I}_j} | (x,y) \in r_{ij}\}$ and $r_{ij}(C)$ to denote $\bigcup_{x \in C} r_{ij}(x)$.

A distributed interpretation $\mathfrak{I}$ satisfies a distributed terminology $\mathfrak{T}$ if:

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

- $\mathcal{I}_i$ satisfies $\mathcal{T}_i$ for all $i \in I$
- $r_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$ for all $i : C \xrightarrow{\sqsubseteq} j : D$ in $\mathfrak{B}_{ij}$
- $r_{ij}(C^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_j}$ for all $i : C \xrightarrow{\sqsupseteq} j : D$ in $\mathfrak{B}_{ij}$

In this case we call $\mathfrak{I}$ a model for $\mathfrak{T}$. A concept $i : D$ subsumes a concept $i : C$ ($i : C \sqsubseteq D$) if for all models of $\mathfrak{T}$ we have $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$.

2.4. Modular ontologies

We can now define our notion of a modular ontology in terms of Distributed Description Logics. In fact, our notion of a modular ontology is a restricted form of distributed terminology as defined above. The restrictions we impose concern the architecture of the distributed terminology as well as the expressiveness of semantic relations between terminologies. These restrictions are motivated by the aims of (1) providing an alternative to the standard notion of import in OWL and (2) the goal of providing support for localized reasoning and maintenance of the modular ontology. In the following, we will first discuss the architecture of a modular ontology and then introduce the restrictions imposed on semantic relations.

2.4.1. Architecture

As described above, DDL makes a clear distinction between terminologies and semantic mappings between them in terms of bridge rules, which in principle are independent of the terminologies. This makes the model quite flexible; for example, it permits having different sets of mapping rules connecting the same set of ontologies. In this way it is possible to encode different views on how the terminologies relate to each other. In contrast, our aim is to enable the use of external knowledge in a terminology similar to the ability of OWL to use concept and role names defined in different terminologies. This view is different from the model of Distributed Description Logics as it makes the semantic links to other models part of the terminology. Being part of the terminology implies that there is only one way of connecting to these external definitions which is assumed to be agreed on by the users of the local terminology.

We achieve this localization of semantic relations by introducing the notion of externally defined concepts in a terminology. We divide the set of concept names in a terminology into internally defined concepts $\mathcal{C}_I$ and externally defined concepts $\mathcal{C}_E$ resulting into the following description of the set of all concept names $\mathcal{C}(\mathcal{C} = \mathcal{C}_I \cup \mathcal{C}_E, \mathcal{C}_I \cap \mathcal{C}_E = \emptyset)$.

We consider externally defined concepts to be concept names linked to a concept expression defined in another terminology using bridge rules. An external concept definition in terminology $\mathcal{T}_i$ is an axiom of the form: $i : C \equiv \mathcal{T}_j : D$ where C is a concept name in $\mathcal{T}_i$, $\mathcal{T}_j$ is a terminology different from the one in which the external concept is defined and D is a concept expression in $\mathcal{T}_j$. Note that although D is syntactically represented in $\mathcal{T}_i$ it actually represents a concept in $\mathcal{T}_j$. In particular the expression D is only allowed to contain concepts defined in $\mathcal{T}_j$. This definition is very close to the OWL mechanism of using concept and role names from other name spaces in definitions.

We give external concept definitions a semantics in terms of distributed description logics by defining external concept definitions to be an alternative notation for a pair of bridge rules:

$$i : C \equiv \mathcal{T}_j : D \Longleftrightarrow j : D \xrightarrow{\sqsubseteq} i : C \wedge j : D \xrightarrow{\sqsupseteq} i : C$$

The correspondence between external concept definitions and bridge rules allows us to base our further investigations on the formal results that have been established for distributed $\mathcal{SHIQ}$ terminologies.

2.4.2. Restricting mapping expressiveness

In Distributed Description Logics, there are no restrictions on the antecedent of a bridge rule—except that it has to be a valid concept of the source terminology. In our framework, we restrict this freedom for the sake of an easier maintenance of the semantic relations between terminologies. This restriction is motivated by our earlier work on keeping integrity in modular ontologies reported in [32]. In that work, we proposed a heuristic approach for determining the impact of changes in other modules on the correctness of local subsumption reasoning. The approach relied on the fact that changes were only monotonically propagated to other modules. In order to achieve this effect also in the framework of distributed description logics, we restrict the language used to specify externally defined concepts to a sublanguage of $\mathcal{SHIQ}$ that does not contain operators that can have a non-monotonic effect, in particular negation, universal restrictions and qualified number restrictions that limit the number of related concepts. More precisely, we allow concept expressions that are defined in the following sublanguage of $\mathcal{SHIQ}$:

$$C, D \rightarrow \top \mid \perp \mid A \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \exists r^{-}.C \mid \geqslant nr.C \mid \geqslant nr^{-}.C$$

In order to restrict the semantic correspondences between terminologies in our model, we now only allow the concept expressions D in the definition of external concepts to be valid concepts over terminology $\mathcal{T}_j$ with respect to the sublanguage defined above. We denote such concepts as $D_{\mathcal{Q}}$ and consider external concept expressions of the form $i : C \equiv j : D_{\mathcal{Q}}$.

3. Reasoning in modular ontologies

The direct correspondence of our framework to Distributed Description Logic allows us to base inference in modular ontologies on known results for the corresponding DDL. In particular, we can provide completeness and complexity results for reasoning in modular ontologies. We extend the existing work on reasoning in DDL with the notion of compilation of implied subsumption relations. Specifically, we use reasoning methods for Distributed Description Logics to derive subsumption relations between externally defined concepts in modules and explicitly add the derived subsumption relations as axioms to the module. The results of [30] guarantee that after this compilation step reasoning can be performed locally without considering other modules unless there are changes in the system.

In the following, we first briefly review basic definitions of reasoning in distributed description logics and prove that it has the same worst-case complexity as reasoning in $\mathcal{SHIQ}$. We then present the compilation of subsumption relations and discuss conditions for completeness and consistency.

3.1. Reasoning based on DDL

Reasoning in DDL differs from reasoning in traditional Description Logics by the way knowledge is propagated between T-Boxes by certain combinations of bridge rules. The simplest case in which knowledge is propagated is the following:

$$\frac{i : A \stackrel{\sqsupseteq}{\rightarrow} j : G, i : B \stackrel{\sqsubseteq}{\rightarrow} j : H, i : A \sqsubseteq B}{j : G \sqsubseteq H} \quad (1)$$

This means that the subsumption between two concepts in a T-Box can depend on the subsumption between two concepts in a different T-Box if the subsumed concepts are linked by the *onto*- and the subsuming concepts by an *into* rule. In languages that support disjunction, this basic propagation rule can be generalized to subsumption between a concept and a disjunction of other concepts in the following way:

$$\frac{i : A \stackrel{\sqsupseteq}{\rightarrow} j : G, i : B_k \stackrel{\sqsubseteq}{\rightarrow} j : H_k (1 \leqslant k \leqslant n), i : A \sqsubseteq \bigsqcup_{k=1}^{n} B}{j : G \sqsubseteq \bigsqcup_{k=1}^{n} H_k} \quad (2)$$

It has been shown that this general propagation rule completely describes reasoning in DDLs that goes beyond well known methods for reasoning in Description Logics. To be more specific, adding the inference rule in Eq. 2 to existing tableaux reasoning methods leads to a correct and complete method for reasoning in DDLs. A corresponding result using a fixpoint operator is given in [30]. Based on these results, we can define a general inference rule for the case of modular ontologies in the following way:

$$\frac{i : A \equiv j : G, i : B_k \equiv j : H_k (1 \leqslant k \leqslant n), i : A \sqsubseteq \bigsqcup_{k=1}^{n} B}{j : G \sqsubseteq \bigsqcup_{k=1}^{n} H_k} \quad (3)$$

There are a number of consequences of this result for reasoning in modular ontologies.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

Correctness and Completeness: From the basic propagation rule, we can see that subsumption between externally defined concepts follows from subsumption of their definitions in the (same) external module. This is because each external concept definition corresponds to an *into* and an *onto* rule between the concept name and its definition. The language we consider is $\mathcal{SHIQ}$ and therefore we have to consider the general propagation rule because we have disjunction in our language. This means that it is not enough to simply check whether subsumption between the definitions of two externally defined concepts in the external module is complete, but we have to consider all subsets of the set of external concepts. We will discuss this point in more detail in the next section.

Complexity: As we reduce reasoning in modular ontologies to reasoning in DDLs with $\mathcal{SHIQ}$ as a local language, complexity results can be derived from known results on reasoning in $\mathcal{SHIQ}$ and Distributed Description Logics.

Theorem 1 (Complexity). *Reasoning in modular ontologies is Exp-Time Complete.*

Proof. We show that reasoning in modular ontologies has the same complexity as reasoning in $\mathcal{SHIQ}$. As reasoning in $\mathcal{SHIQ}$ is Exp-Time Complete [33], this establishes the result.

- Reasoning in modular ontologies is at least as hard as reasoning in $\mathcal{SHIQ}$: in the extreme case a modular ontology consists only of a single module without external concepts, thus reasoning in modular ontologies is equivalent to reasoning in $\mathcal{SHIQ}$.
- Reasoning in modular ontologies is not harder than reasoning in $\mathcal{SHIQ}$ because we reduce reasoning in modular ontologies to reasoning in DDLs. There exists a reduction of reasoning in DDLs with $\mathcal{SHIQ}$ as a local language to $\mathcal{SHIQ}$ [30]. Both reductions are linear in the size of the resulting terminology and therefore do not change the complexity class.

This result shows that the complexity of reasoning in modular ontologies is not worse than reasoning in the web ontology language. Using the reduction of DDL to $\mathcal{SHIQ}$ it is even possible to use existing OWL reasoners for reasoning with modular ontologies. Although practical implementations of OWL reasoners have shown that good average case performance can be achieved, the worst case complexity is still very high and asks for further optimization.

3.2. Compilation and integrity

Existing reasoners for expressive Description Logics are highly optimized with respect to deciding subsumption in the context of a single T-Box. Serafini and Tamilin present a distributed reasoning system that extends existing reasoners to distributed T-Boxes [34]. In theory, this system is complete with respect to the propagation rules described above and has—as we have argued—the same worst-case complexity. In practice, however, reasoning with multiple, possible distributed modules, brings some new problems with respect to completeness and reasoning performance. First of all the completeness of the distributed reasoners depends on the availability of local reasoners for all T-Boxes in the system. In a loosely coupled network without central control this cannot always be guaranteed as network nodes can be unreachable or even leave the network. In this case, necessary subsumption tests cannot be performed at these nodes leading to a possible incompleteness. Another problem currently not addressed in the work of Serafini and Tamilin are performance problems due to communication costs between the different nodes in the system. Work in the area of distributed databases has shown that communication costs often become serious bottlenecks in distributed systems.

In order to overcome these problems we propose to compute subsumption relations between external concepts offline and store them as explicit axioms in the local ontologies. If we compute these relations using the reasoner mentioned above we have the guarantee that reasoning about subsumption in each module can be done without caring about the availability of other nodes in the network. This also has the advantage that no communication costs occur as part of online reasoning.

Of course these runtime benefits have their price in terms of computational complexity of the compilation step. The completeness of the propagation rule given in Eq. 2 tells us that to be independent from other mod-

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

ules we only have to consider subsumption relations between externally defined concepts, as only such subsumption relations can be propagated from outside. What we have to check is subsumption between each external concept and the disjunction of all combinations of other external concepts. For a local module, this process is defined in Algorithm 1.

Algorithm 1. Compile

Require: An T-Box $\mathscr{T}$ with external concepts $\mathscr{C}_E$
forall $c \in \mathscr{C}_E$ **do**
  candidates $:= \mathscr{P}(\mathscr{C}_E - \{c\})$
  forall $d \in$ candidates **do**
    If $\mathscr{I} \models c \sqsubseteq \bigsqcup_{e \in d} e$ **then**
      $\mathscr{T} := \mathscr{T} \cup \{c \sqsubseteq \bigsqcup_{e \in d} e\}$
    end if
  end for
end for

If we denote the number of external concepts $\mathscr{C}_E$ as n, the worst-time complexity of the compilation method is $O(n \cdot 2^{(n-1)})$ as can easily be seen from the algorithm. As deciding the subsumption relation in the conditional statement of the algorithm itself is already Exp-Time Complete and this test has to be carried out an exponential number of times with respect to the number of external concepts, compiling all implied statements is computationally very expensive. We therefore do not want to perform the compilation step more often than absolutely necessary to guarantee that local reasoning is still complete.

While the results of Serafini and others [34] guarantee that local reasoning is correct and complete at the time the compilation is carried out, a problem occurs when changes are made to the system. Changes in the definitions of the external concepts, but also changes in the definitions of concepts and relations in other modules can make local reasoning incomplete or inconsistent. In order to prevent situations in which local reasoning is not correct and complete any more we introduce the notion of integrity of a modular ontology.

Definition 1 (*Integrity*). Let $\mathfrak{T} = (\{\mathscr{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I})$ be a modular ontology with interpretation $\mathfrak{I} = (\{\mathscr{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I})$ then we say that integrity holds for $\mathfrak{T}$ if for all $\mathscr{T}'_i = compile(\mathscr{T}_i)$ with interpretation $\mathscr{I}'_i$ we have:

$$\mathscr{I}'_i \models C \sqsubseteq D \Longleftrightarrow \mathfrak{I} \models i : C \sqsubseteq D$$

for any pair of legal concept expressions C and D in $\mathscr{T}'_i$.

The notion of integrity gives us a criterion for deciding whether compiled results are still valid. What the definition does not provide is an operational account for checking it. A direct use of the definition would involve a complete check of all derivable subsumption relations. As we have argued above this approach is extremely expensive. In the following, we therefore present a heuristic approach for checking integrity in modular ontologies that is driven by changes made to the ontology. The approach is capable of determining situations in which changes to a modular ontology do not affect integrity and therefore no re-compilation is necessary.

4. Evolution management

As we have argued above, guaranteeing integrity of compiled subsumption relations is the main problem in modular ontologies. In principle, all compiled subsumption relations have to be recomputed to test whether they are still valid in the given state of the system. This of course means sacrificing the advantages of the compilation approach in terms of local reasoning and reduced complexity. Fortunately, we can do better than checking all compiled axioms each time we perform reasoning.

The first possible improvement is to move away from an active checking for changes towards a mechanism where each local module remembers and records changes made to it. We can also think of a system where

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

individual modules actively notify other modules of changes to its local knowledge. This frees us from doing a complete check of the compiled knowledge and allows us to concentrate on these parts of the knowledge that actually were subject to changes.

The second improvement is in terms of an analysis of the impact a change in another module actually has on compiled knowledge. This is important as in real-world scenarios it turns out that a large part of the changes do not really affect the logical theory but are rather changes to the syntactic representation or changes in the naming of concepts and relations. While the latter have to be propagated to the definitions of the external concepts, they do not actually affect the compiled subsumption relations. Further, even if the logical theory underlying the ontology is affected by a change, this does not mean that it affects the compiled subsumption relations. This means that we have to find ways to distinguish changes that do have an impact on compiled relations from those that do not have an impact. In fact, the choice to restrict the language admissible in the definitions of external concepts allows us to precisely characterize these kinds of changes.

In the following, we concentrate on the analysis of the impact of changes on the validity of compiled subsumption relations. We first give a characterization of harmless and harmful changes. Here harmless changes are those that do not have an effect on compiled subsumption relations. Harmful changes are changes that do have a potential influence on compiled knowledge. We present mechanisms for classifying changes as harmless or harmful based on a syntactic analysis of changes made to an ontology. Finally, we present a simple mechanism that uses this information to decide whether the knowledge in a module has to be re-compiled.

4.1. Determining harmless changes

As compiled knowledge reflects subsumption relations between a query concept and a disjunction of other query concepts a harmless change is a set of modifications to an ontology that does not change these subsumption relations. Finding harmless changes is therefore a matter of deciding whether the modifications affect the subsumption relation between a query concept and a disjunction of other query concepts. It is quite obvious that a complete decision procedure for this problem has the same complexity as general subsumption reasoning in the modular ontology and does therefore not improve the situation. For this reason, we propose a sound but incomplete method that abstracts from the detailed definition of concepts and uses the semantic relation between the old and the new version of a concept in the following way.

The method considers every concept and relation in an ontology module that has been subject to a change. Assuming that C represents the concept under consideration before and C' the concept after the change, there are four ways in which the old version C may semantically relate to the new version C':

(1) the meaning of a concept is not changed: $C \equiv C'$ (e.g. because the change was in another part of the ontology, or because it was only syntactical);
(2) the meaning of a concept is changed in such a way that the concept becomes more general: $C \sqsubseteq C'$;
(3) the meaning of a concept is changed in such a way that the concept becomes more specific: $C' \sqsubseteq C$;
(4) the meaning of a concept is changed in such a way that there is no subsumption relationship between C and C'.

We can define the semantic relation between different versions of the same concept based on the set of possible interpretations of the old and the new ontology in the following way.

Definition 2 (*Semantics of Change*). Let C and C' be two version of the same concepts in ontologies O and O' respectively, then we say that C is more general than C' ($C \sqsupseteq C'$) if and only if for all possible interpretations $\mathscr{I}$ and for all $x \in \Delta^{\mathscr{I}}$ we have $\mathscr{I}, O \models x \in C^{\mathscr{I}}$ implies $\mathscr{I}, O' \models x \in C'^{\mathscr{I}}$. Analogously, we say that C is more specific that $C'(C \sqsubseteq C')$ if an only if for all possible interpretations $\mathscr{I}$ and for all $x \in \Delta^{\mathscr{I}}$ we have $\mathscr{I}, O' \models x \in C'^{\mathscr{I}}$ implies $\mathscr{I}, O \models x \in C^{\mathscr{I}}$.

The same list holds for relations that are subject to change. The next question is how these different kinds of semantic relations between the old and the new version of a concept influences compiled knowledge. In order to understand this influence, we have to look at the influence of changes on the interpretation of query con-

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

cepts. We take advantage of the fact that there is a very tight relation between changes in concepts of the external ontology and implied changes to the query concepts using these concepts:

Lemma 1 (monotonicity of effect). *Let $C \equiv \mathcal{T}_j : Q$ an external concept expression. Let $c(Q)$ be the set of all concept names and $r(Q)$ the set of all relation names occurring in Q, let further $C \in c(Q)$ and $R \in r(Q)$ then changing C has the same impact on the interpretation of Q as it has on the interpretation of C, in particular, we have $C \sqsubseteq C' \Rightarrow Q \sqsubseteq Q'$ and $C' \sqsubseteq C \Rightarrow Q' \sqsubseteq Q$ where Q' is the query as being interpreted after changing C. Analogously, a change of R has the same effect on Q.*

Proof Sketch We prove lemma 1 by structural induction over expressions in the sublanguage defined in Section 2.4.2. For the *induction basis* $Q = A$ the lemma trivially holds. In the induction step, we prove for every operator that the whole expression becomes more general (specific) if either a concept or a relation occurring in the expression becomes more general (specific). For conjunction and disjunction this directly follows from their correspondence to set operations on the interpretation domain. It remains to be shown that lemma 1 also holds for expressions of the form $(\geqslant nR.C)$ and $(\geqslant nR^{-}.C)$ (existential quantifiers are a special case). For $(\geqslant nR.C)$ and $C \sqsubseteq C'$ this holds because all R-successors of C in $C^{\mathcal{I}}$ are also in $C'^{\mathcal{I}}$. Therefore we have $(\geqslant nr.C')^{\mathcal{I}} \supseteq (\geqslant nr.C)'^{\mathcal{I}}$. Further, there are no R-successors in $C'^{\mathcal{I}}$ that are not in $D^{\mathcal{I}}$. Therefore we have $(\geqslant nr.C)'^{\mathcal{I}} \subseteq (\geqslant nr.C)^{\mathcal{I}}$. The same argument holds for r^{-}. For $R \sqsubseteq R\prime$ the argument is similar. This time, the subsumption follows from the fact that there are less members of C that are potentially in the relation R with objects in Q'.

We can exploit this relation between the interpretation of external concepts and the concept names in their definitions in order to identify the effect of changes in the external ontology on the subsumption relations between different query concepts. First of all, the above result directly generalizes to multiple changes with the same effect, i.e. a query Q becomes more general (specific) or stays the same if none of the elements in $c(Q) \cup r(Q)$ become more specific (general). Further, the subsumption relation between an external concept C and the disjunction of other external concepts does not change if all concepts in the disjunction become more general or if the concept C becomes more specific. Combining these two observations, we derive the following characterization of harmless change.

Theorem 2 (harmless change). *Let $C_0 \equiv \mathcal{T}_j : D_0, C_1 \equiv \mathcal{T}_j : D_1, \ldots, C_m \equiv \mathcal{T}_j : D_m$ be external concept definitions such that $\Im \models C_0 \sqsubseteq C_1 \sqcup \ldots \sqcup C_m$, then a change is harmless with respect to the subsumption relation above if*:

- $X' \sqsubseteq X$ for all $X \in c(D_0) \cup r(D_0)$,
- $X' \sqsupseteq X$ for all $X \in c(D_i) \cup r(D_i)$, $i = 1, \ldots, m$

Note again that the implication does not hold in the opposite direction.

The theorem provides us with a correct but incomplete method for deciding whether a change is harmless given that we know the semantic relation between the old and the new definition of concepts and relations that were subject to changes. This method is a very basic version of the underlying idea of assessing the impact of changes. We can think of more complete versions of the method that use a deeper analysis of the structure of the concept expressions involved. Our experiences are, however, that this basic heuristic already covers most cases that occur in practice, especially, because the definition above includes cases where most of the concepts stay unchanged.

4.2. Characterizing changes

Now that we are able to determine the consequence of changes in the concept hierarchy on the integrity of the mapping, we still need to know what the effect of specific modifications on the interpretation of a concept is (i.e. whether it becomes more general or more specific). As our goal is to determine the integrity of mappings without having to do classification, we describe what theoretically could happen to a concept as result of a

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

Table 2
Some modifications to an ontology and their effects on the classification of concepts in the hierarchy

Operation	Effect
Add a role restriction to concept *C*	*C*: Specialized
Complex: change the superconcept of concept *C* to a concept lower in the hierarchy	*C*: Specialized
Complex: restrict the range of a role *R* (*effect on all C that have a restriction on R*)	*R*: Specialized, *C*: Specialized
Remove a superconcept relation of a concept *C*	*C*: Generalized
Change the concept definition of *C* from primitive to defined	*C*: Generalized
Add a concept definition *A*	*C*: Unknown
Complex: add a (not further specified) subconcept *A* of *C*	*C*: No effect
Define a role *R* as functional	*R*: Specialized

modification in the ontology. To do so, we have listed all possible change operations to an ontology according to the OWL[2] knowledge model in the same style as done in [35]. The list of operations is in principle extendable to other knowledge models.

The list of change operations consists of two types of operations: (1) *atomic change operations*, such as *add range restriction* or *delete subconcept relation* and (2) *complex change operations*, which consist of multiple atomic operations and/or incorporate some additional knowledge. Complex changes are often more useful to specify effects than the atomic changes, as they incorporate some of the semantic consequences. For example, for operations like *concept moved up*, or *domain enlarged*, we can specify the effect more accurately than for the atomic operations *superconcept changed* and *domain modified*.[3] Atomic changes can be detected at a structural level, i.e. by comparing the old and new definition of a concept, and are therefore computationally cheap with a linear complexity. To identify complex changes, we also need to take some of the semantic relations in the ontology into account. This makes the complexity of the identification of complex changes potentially as bad as determining subsumption in $\mathcal{SHIQ}$, i.e. Exp-Time Complete. However, in practice many complex changes can be detected at a structural level, e.g. by looking at explicitly stated subclass relations.

Table 2 contains some examples of operations and their effect on the classification of concepts. The table only shows a few examples, although our full ontology of change operations contains around 120 operations. This number is not fixed, as new complex changes can be defined. A snapshot of the change ontology can be found online.[4] The specification of effects is not complete, in the sense that it describes "worst-case" scenarios, and that for some operations the effect is "unknown" (i.e. unpredictable). If the method's output is "unknown" this means that in order to determine the semantic relation between the two versions we would have to perform logical reasoning. In contrast to [37] who provide complete semantics of changes, we prefer to use heuristics in order to avoid expensive reasoning about the impact of changes. By restricting the change detection to changes that can be detected at a structural level, the complexity of our change detection heuristic is linear.

4.3. Update management

With the elements that we described in this section, we now have a complete procedure to determine whether compiled knowledge in an ontology module is still valid when the external ontology modules are changed. The complete procedure is as follows. For each external concept *C*:

(1) determine the changes that are performed in the external ontology (e.g. by using the record of changes);
(2) heuristically determine the effect of the changes on the interpretation of the concepts and relations (where multiple changes to a concept or relation that have an opposite effect lead to the effect "unknown");

2 See http://www.w3.org/TR/owl-features/.
3 For a complete list, see [36].
4 http://ontoview.org/changes/2/1.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

(3) create a list of all concept and relation names that occur in the external concept expression D for each concept C in the subsuming parts of the implied subsumption relations;
(4) check whether all concepts and relations in this list remained unchanged or became more general;
(5) create a list of all concept- and relation names that occur in the external concept expression D for each concept C in the subsumed parts of the implied subsumption relations;
(6) check whether all concepts and relation in this list are unchanged or became more specific.

In cases where we cannot guarantee that integrity is preserved, we recompute and re-compile the implied subsumption statements. We thus restore integrity and make correct local reasoning possible.

Algorithm 2. Update

```
Require: Ontology Module M
Require: Ontology Module M_j
  forall compiled axioms C_1 ⊑ D_1 ⊔ ··· ⊔ D_m in M^c do
    forall X ∈ c(C) ∪ r(C) do
      if effect on X is 'generalized' or 'unknown' then
        M^c := Recompile(M, M_j)
      end if
    end for
  for all D_i, i = 1, ..., m do
    forall X ∈ c(D_i) ∪ r(D_i) do
      if effect on X is 'specialized' or 'unknown' then
        M^c := Recompile(M, M_j)
      end if
    end for
  end for
end for
```

We describe the procedure in a more structured way in Algorithm 2. The algorithm triggers a (re-)compilation step only if it is required in order to resume integrity. Otherwise no action is taken, because the previously compiled knowledge is still valid. In principle, all the steps can be automated. A tool that helps to automate steps (1) and (2) is described in [38]. This tool will compare two versions of an ontology and derive the list of change operations that is necessary to transform the one into the other.

The worst-time complexity of the update procedure once the effects of changes are known is linear in the number of concept and relation names occurring in compiled subsumption statements (in the worst case the effect of a change on every concept and relation name in the compiled axioms has to be checked). In practice, we expect the number of compiled axioms to be relatively small leading to a quite efficient procedure.

5. Application in a case study

In order to support the claims made about the advantage of modular ontologies, we applied our model in a small case study that has been carried out in the course of the WonderWeb project.[5] Our main intention was to show that the update-management procedure presented in the last section can be used to avoid the computation of subsumption relations in many cases. For this purpose, we defined a small example ontology using mappings to an ontology in the human resource (HR) domain. We used the changes that occurred in the HR ontology during the different steps of the case study and determined the impact on our example ontology. Besides this, the case study provides us with examples of implied subsumption some of which are non-trivial but likely to occur in real-life situations.

[5] See http://wonderweb.semanticweb.org.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

5.1. The WonderWeb case study

WonderWeb is a EU/IST project that ran from 2002 till 2004. The aim of the project was to develop and demonstrate the infrastructure required for the large-scale deployment of ontologies as the foundation for the Semantic Web. In the context of this project the *Descriptive Ontology for Linguistic and Cognitive Engineering* (DOLCE) [39] has been developed. DOLCE is the first module in a library of foundational ontologies. The taxonomy of the most basic categories consists of concepts such as "location", "social agent", "event", "par-

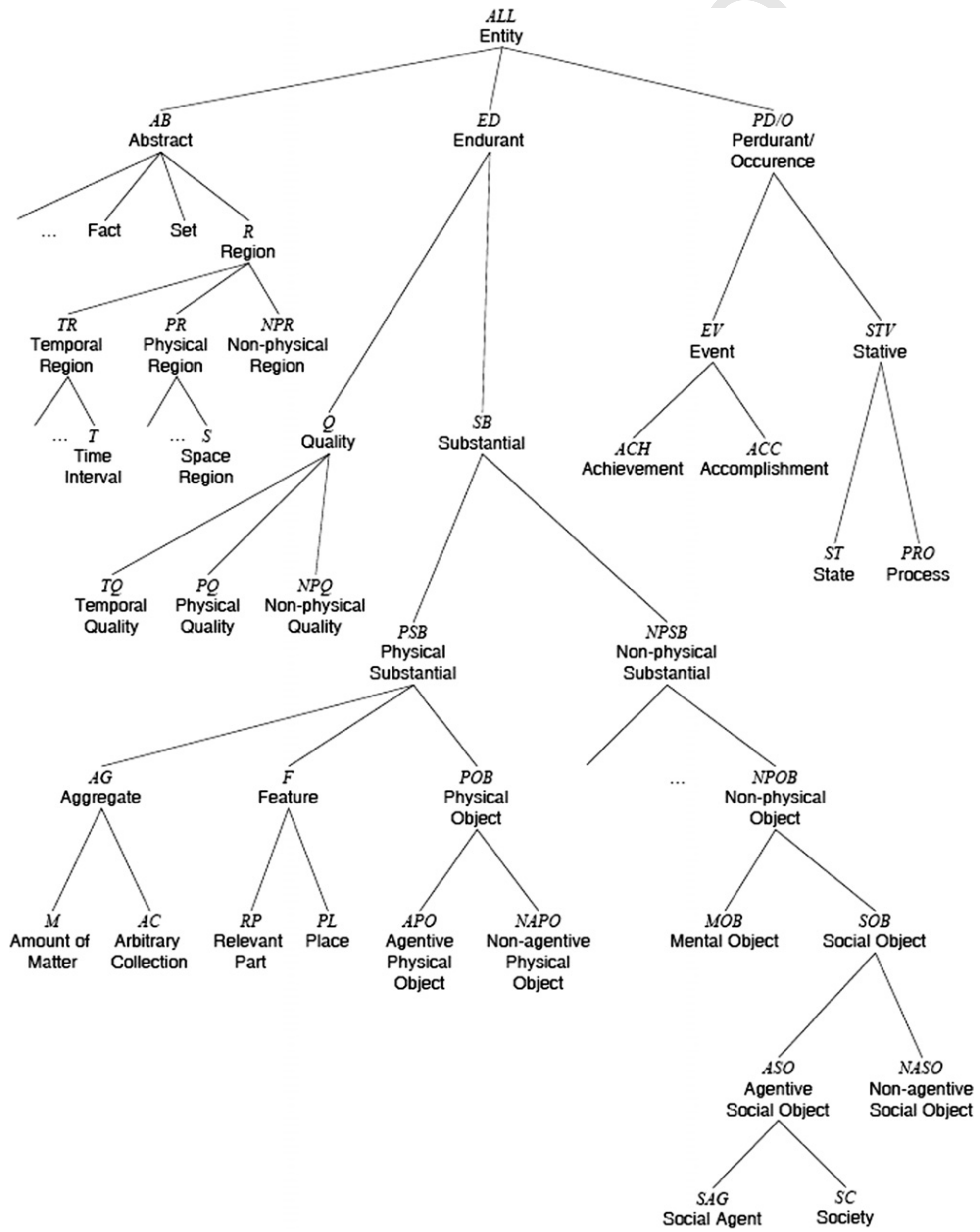

Fig. 1. Parts of the DOLCE upper level ontology (taken from [39]).

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

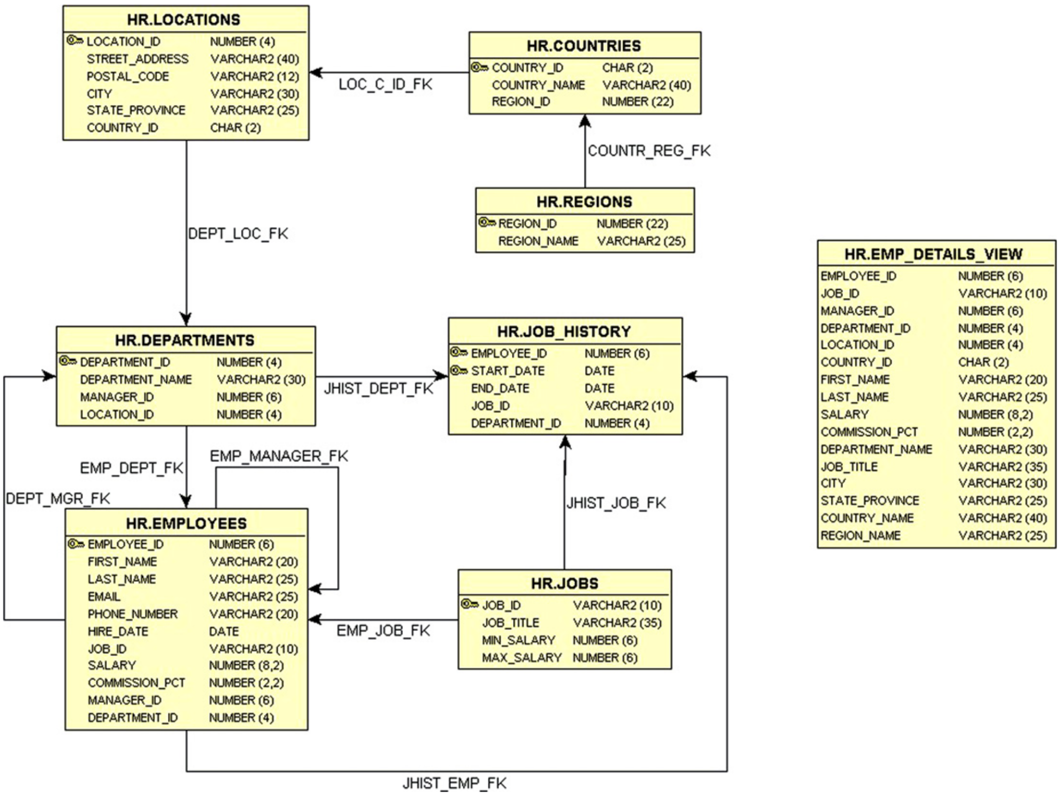

Fig. 2. Schema of the human resource database.

ticipant'' and ''region''. Fig. 1 shows the core part of the DOLCE ontology, the complete DOLCE ontology can be found online.[6]

One of the roles of foundational ontologies in general and DOLCE in particular is to clarify hidden assumptions underlying existing ontologies or linguistic resources by manually mapping existing categories into the categories defined in the foundational ontology [40]. Inconsistencies in the combined ontology point to modeling errors or wrong assumptions in the original ontology. Solving the inconsistencies will improve the quality of the initial ontology. The methodology around DOLCE describes a three step procedure for mapping existing ontologies to DOLCE: alignment, refinement and tidying.

The case study that has been carried out in the project integrates different methods in the life-cycle of an ontology that is used on the Semantic Web, i.e. ontology creation, ontology refinement and ontology deployment. The case study starts from an existing database schema in the human resource (HR) domain. The initial database schema is shown in Fig. 2. A first version of an ontology is created by a tool that automatically converts a database schema into an ontology [41]. In the next phase, the quality of the ontology is improved by manually mapping this ontology to DOLCE. First, the HR ontology is aligned with DOLCE, and in several successive steps the resulting ontology is further refined. During this process, the ontology changes continuously, which causes problems when other ontologies refer to definitions in the evolving ontology.

The original HR ontology combined with DOLCE is referred to as the DOLCE + HR ontology. In order to demonstrate the update management procedure, we created another ontology (which we call the *local ontology*) that uses terms and definitions from the evolving DOLCE + HR ontology (the *external ontology*). The

[6] The main file can be found at http://www.loa-cnr.it/ontologies/DOLCE-Lite.owl, other files are referred from this file.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

local ontology defines the concept *FulltimeEmployee* with a superconcept *Employee* and two subconcepts *DepartmentMember* and *HeadOfDepartment*, using terms from the DOLCE + HR ontology.

The specific problem in our case is that the changes in the DOLCE + HR ontology could affect the reasoning in the local ontology. We want to be able to predict whether or not the reasoning in the local ontology is still valid for specific changes in the external ontology.

The evolution of the DOLCE + HR ontology consisted of several steps. Each of these steps involves some typical changes. We will briefly summarize them and show some changes that are typical for a specific step. In Section 5.3, we discuss the effect of the changes described here on the integrity of implied subsumption relations in the local ontology.

- In the first step, the extracted HR ontology is aligned with the DOLCE foundational ontology, i.e. the concepts and roles in the HR ontology are connected to concepts and roles in the DOLCE ontology via subsumption relations. For example, the concept *Departments* from the HR ontology is made a subconcept of *Social-Unit* in DOLCE.
- The refinement step involves a lot of changes. Some role restrictions are added, and some additional concepts and roles are created to define the HR concepts more precisely. For example, the concept *Administrative-Unit* is introduced as a new subconcept of *Social-Unit*, and the concept *Departments* is made a subconcept of it. Also, the range of the role *email* is restricted from *Abstract-Region* to its new subconcept *Email* and the range of *manager-id* is specialized by restricting its range from *Employee* to the subconcept *Manager*.
- In the next step, a number of concepts and roles are renamed to names that better reflect their meaning. For example, *Departments* is renamed to *Department* (singular), and the two different variants of the role *manager-id* are renamed to *employee-manager* and *department-manager*.
- In the final step, the tidying step, all roles and concepts that are not necessary any more are removed and transformed into role restrictions. For example, the role *employee-email* is deleted from *Employee* and replaced by an existential restriction in the concept *Person* on the role *abstract-location* to the concept *Email*.

5.2. Modularization in the case study

If we now consider the modularization in the case study, we have a local ontology with a concept hierarchy that is built up by the following explicitly stated subsumption relations:

$$FulltimeEmployee \sqsubseteq Employee$$
$$DepartmentMember \sqsubseteq FulltimeEmployee$$
$$HeadOfDepartment \sqsubseteq FulltimeEmployee$$

This ontology introduces *FulltimeEmployee* as a new concept, not present in the case study ontology. Consequently, this concept is only defined in terms of its relation to other concepts in the local ontology.

All other concepts are externally defined in terms of ontology based queries over the case study ontology. The first external definition concerns the concept *Employee* that is equivalent to the *Employee* concept in the case study ontology. This can be defined by the following trivial view:

$$Employee \equiv HR : Employee$$

Another concept that is externally defined is the *HeadOfDepartment* concept. We define it to be the set of all instances that are in the range of the *department-manager* role. The definition of this view given below shows that our approach is flexible enough to define concepts in terms of relations

$$HeadOfDepartment \equiv HR : \exists department\text{-}manager.\top$$

An example of a more complex external concept definition is the concept *DepartmentMember*, which is defined using a query that consists of three conjuncts, claiming that a department member is an employee that is in the *department-member* role with a department

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

$$DepartmentMember \equiv HR : Employee \sqcap \exists department^{-}member^{-}.Department \quad (4)$$

5.2.1. Implied subsumption relations

To allow for local reasoning, we need to determine the implied subsumption relations. If we now consider logical reasoning about these external definitions, we immediately see that the definition of *Employee* subsumes the definition of *DepartmentMember*, as the former occurs as part of the definition of the latter.

$$\models DepartmentMember \sqsubseteq Employee \quad (5)$$

At a first glance, there is no relation between the definition of *HeadOfDepartment* and the two other statements as it does not use any of the concept or role names. However, when we use the background knowledge provided by the case study ontology we can derive some implied subsumption relations. The reasoning is as follows. Because the range of *department-manager* is set to *Employee* and the domain to *Department*, the definition of *HeadOfDepartment* is equivalent to:

$$Employee \sqcap \exists department - manager^{-}.Department$$

As we further know that *department-manager* is a subrole of *department-member*, we can derive the following subsumption relation between the externally defined concepts:

$$\models HeadOfDepartment \sqsubseteq Employee \quad (6)$$
$$\models HeadOfDepartment \sqsubseteq DepartmentMember \quad (7)$$

Besides the subsumption relations between the external concepts themselves, we also need to determine the subsumption relations between the external concepts and the disjunction of all combinations of other external concepts. For our example, with $HeadOfDepartment \sqsubseteq DepartmentMember \sqsubseteq Employee$, this results in the following relations:

$$\models Employee \sqsupseteq HeadOfDepartment \sqcup DepartmentMember \quad (8)$$
$$\models HeadOfDepartment \sqsubseteq Employee \sqcup DepartmentMember \quad (9)$$
$$\models DepartmentMember \sqsubseteq Employee \sqcup HeadOfDepartment \quad (10)$$

When the relations (5)–(10) are added to the local ontology, it possible to do subsumption reasoning without having to access the DOLCE + HR ontology any more.

5.3. Updating the models

We will now illustrate that the conclusions of the procedure are correct by studying the impact of some changes mentioned in the bullet list in Section 5.1. We do not cover all changes, but discuss three typical cases. For the other changes, a similar analysis could be done.

5.3.1. Example 1: The employee concept

For the *Employee* concept, the last step resulted in the removal of the *employee-email* relation. Our rules tell us that this change makes the new version more general compared to its old version:

$$Employee \sqsubseteq Employee'$$

According to our procedure, this should not be a problem because employee is in the "subsuming list".

When we analyze this change, we see that it has an impact on the definition of the concept *DepartmentMember* as it enlarges the set of objects allowed to take the first place in the has-member relation. This leads to a new definition of *DepartmentMember* with $DepartmentMember \sqsubseteq DepartmentMember'$. As *DepartmentMember* was already more general than *HeadOfDepartment* and the *Employee* concept is not used in the definition of the latter, the implied subsumption relation indeed still holds.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

5.3.2. Example 2: The department-manager relation

In the second example, we have to deal with a change affecting a relation that is used in an external definition. The relation *department-manager* is specialized by restricting its range to the concept *Manager* (which is a subconcept of *Employee*) making it a subrelation of its previous version (i.e. more specific):

$$department\text{-}manager \sqsupseteq department\text{-}manager'$$

According to our procedure, this change is harmful as *department-manager* is in the "subsuming list".

Calculation and an analysis proves that this change indeed has impact on the definition of the concept *HeadOfDepartment*. Because the old version of the role *department-manager* is more general than the new one, it cannot be concluded that the old role (which is used in the external concept expression) is still a subrole of the *department-member* relation. Therefore, the implied subsumption relation *HeadOfDepartment* $\sqsubseteq$ *DepartmentMember* no longer holds. We have to recompute and compile the implied subsumption relations in order to guarantee integrity.

5.3.3. Example 3: The department concept

For the *Department* concept, different changes happened. For example, it has been made a subconcept of *Social-Unit*, it has been renamed from *Departments* and a relation has been removed. These changes have a contradictory effect, according to our heuristics, and as a consequence we cannot predict the relation between the old and the new versions. In this case our current heuristics leave us with no other option than recomputing the implied subsumption relations.

6. Summary

In this article, we discussed an infrastructure for the representation of and reasoning with modular ontologies. The intention was to enhance the existing semantic web infrastructure with notions of modularization that have been proven useful in other areas of computer science, in particular in software engineering. We defined a set of requirements for modular ontologies that arise from expected benefits such as enhanced reuse and more efficient reasoning. Taking the requirements of loose coupling, self containment and integrity as a starting point, we defined a framework for modular ontologies providing the following contributions to the state of the art in ontology representation for the semantic web:

(1) We presented a formal model for describing dependencies between different ontologies. We proposed conjunctive queries for defining concepts using elements from another ontology and presented a model-based semantics in the spirit of Distributed Description Logics that provides us with a notion of logical consequence across different ontologies.
(2) We compared our model with the existing standard, i.e. the web ontology language OWL and showed that the OWL import facilities can easily be captured as a special case in our model. We further showed that our model provides additional expressiveness in particular with respect to modeling relations. In order to get a better idea of the improvements of our model over OWL, we investigated the formal properties of inter module mappings, their impact on reasoning and their intuition.
(3) We described a method for detecting changes in an ontology and for assessing their impact. The main feature of this method is the derivation of conceptual changes from purely syntactic criteria. These conceptual changes in turn provide input for a semantical analysis of the effect on dependent ontologies, in particular on the validity of implied subsumption relations. We applied the method in a case study in the Wonder Web project and were able to determine the impact of changes without logical reasoning.

7. Discussion

There are three major questions connected to the approach for reasoning and managing change in modular ontologies proposed in this article. The first is *feasibility* in terms of computational complexity. As mentioned above, we use a heuristic approach to tackle this problem, which raises the question about the *adequacy* of the heuristics used. We argued that we chose a trade-off that works well in the context of OWL and typical semantic web applications. This focus on a particular kind of representation finally raises the question of *generality* of the approach. We discuss these three basic questions in the following.

7.1. Feasibility

Reasoning in modular ontologies is complex. We have shown that the complexity is essentially the same as for reasoning in Classical Description Logics which are the basis for OWL. We cannot escape this complexity, but we can move parts of the reasoning effort offline by compiling implied subsumption relations as described in Section 3.2. This approach, however, is only feasible if there are phases where the offline computation necessary to compile the implied relations can be done without affecting the performance of the system. Typically, such computations are done 'overnight' when the system load can be assumed to be low. An alternative for situations where this approach is not possible is to do the compilation on the fly. In particular, we can compile implied subsumption relations whenever they are computed in order to answer a user query to the system. This kind of 'lazy compilation' has the advantage that the enormous effort for compiling implied knowledge is done as part of the normal reasoning process. In the beginning, users will not benefit much from this approach, but the time savings increase with each query answered. In this way, we also prevent the compilation of knowledge that is never used.

The main problem connected with the compilation approach, which is also a central aspect of this paper, is the integrity of the compiled knowledge. In general compilation approaches only pay off if the computation time saved by being able to use compiled knowledge is not larger than the effort of updating the compiled knowledge. This means that compilation only makes sense in rather stable systems. In principle, we can assume that knowledge on the terminological level as it is represented in ontologies is normally more stable than instance data as normally found in databases. While changes to ontologies will occur less frequently they can still have a significant impact on the system. For this reason, our work focussed on heuristics for efficiently updating the system when changes occur. In this context, the feasibility of the approach relies on the adequacy of the heuristics chosen.

7.2. Adequacy

Our method for detecting harmful changes is a conservative one. We basically identify changes that are obviously not harmful and refrain from updating when only such changes occurs. The criteria used for this purpose are rather weak as we do not consider the specific changes made to a concept but restrict our analysis to the semantic relation between the old and the new version of the concept definition. The advantage of this choice is the efficiency of the approach as the criterion for harmless changes can be checked in linear time with respect to the number of concept names involved. We further weaken the method by not actually computing the semantic relation between the old and the new version of the concept, but rather determine the relation based on a set of change operations determined by syntactic analysis. This method also constitutes an incomplete heuristic as, for some changes, we cannot determine the relation between the old and the new version. Again, the advantage of this choice lies in the efficiency of the approach, because it allows us to live completely without Description Logic reasoning.

The question that arises is whether the degree of incompleteness of our approach is justified. Currently there are no experimental results that support the usefulness of the heuristics, however, it is clear that the heuristics cover a wide range of relevant cases. On the level of determining the semantic relation between the versions, the library of change operations covers all possible change operations and most of them have a known effect that can be exploited in our approach. The cases in which the effect is not known can be assumed to be the hard cases that will always require complete reasoning independent of the heuristics used. For this reason, we believe that we cannot be much better on this level without falling back to classical subsumption reasoning. On the level of determining harmless change, the heuristics used are quite weak as well. In particular we claim that a change is only harmless if all concepts and relations satisfy certain properties. This can certainly be relaxed if we invest more time in the analysis of the actual definitions of the concepts involved. For example, changes in concepts that occur in both the subsumed and the subsuming concepts are not relevant due to the

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

monotonicity of the effect. We could also try to determine which parts of the definitions actually contribute to the subsumption proof and consider changes in other parts of the definitions as harmless as well. These improved heuristics still fit into the approach proposed and are subject of future work. For the time being, we conclude that the general mechanisms are in place and that the heuristic described in this paper already covers many relevant cases as we show in the examples from the WonderWeb case study. In principle, the choice of the best heuristic will rely on the specific application scenario and in particular on the actual expressiveness used in the models.

7.3. Generality

A final point for discussion is the generality of the approach described. Throughout this paper, we based our discussions on Description Logics as a representation language for ontologies, Distributed Description Logics for providing the semantics of mappings as well as the equivalent of conjunctive queries for describing relations between different modules. All of these choices are carefully made and are motivated by practical as well as theoretical considerations. Probably the most uncontroversial choice is that of Description Logics for encoding ontologies. In the context of semantic web research, Description Logics have become the primary language for describing terminological knowledge mostly in terms of the Web Ontology Language OWL. Our approach covers most of the expressiveness of OWL-DL with the exception of nominals. As a result, most existing OWL ontologies will fit in our framework and could easily be turned into modular ontologies by adding external concepts.

A choice that is less obvious is Distributed Description Logics as a basis for the semantics of mappings. In a recent survey, we compared different approaches for describing mapping semantics [42]. One result of this comparison was that Distributed Description Logics provide the highest degree of de-coupling between different T-Boxes. This is important for our purposes as we want to support localized reasoning. A generalization of Distributed Description Logics in terms of a distributed version of first order logic has been described by Serafini and Ghididi [43]. We could have chosen this more general framework as the basis for our work, however, the drawback of this is the lack of existing reasoning methods. Distributed Description Logics come with a well investigated and implemented proof system that can be used to implement our approach.

The most controversial choice is to restrict the language that can be used to define external concepts. The framework of Distributed Description Logic allows us to use arbitrary $\mathcal{SHIQ}$ expressions in the definitions. A corresponding more general approach would have the same properties with respect to logical consequence, compilation and local reasoning. The restriction to the equivalent of conjunctive queries was motivated by the importance of the monotonicity property for the definition of update heuristics. This means that external concepts can be defined using a more expressive language. This, however, would come at the price that implied subsumption relations concerning this concept would have to be recomputed every time a change occurs. We believe that the restriction proposed in this paper is reasonable as it allows the use of update heuristics and also resembles view-based information integration, the dominant approach for describing mappings between database schemata.

References

[1] S. Bechhofer, I. Horrocks, C. Goble, R. Stevens, OilEd: a reason-able ontology editor for the semantic web, in: F. Baader, G. Brewka, T. Eiter (Eds.), KI 2001: Advances in Artificial Intelligence, Springer, 2001, pp. 396–408.
[2] N. Noy, R. Fergerson, M. Musen, The knowledge model of protege-2000: combining interoperability and flexibility (2000). URL <citeseer.nj.nec.com/noy01knowledge.html>.
[3] I. Horrocks, The FaCT system, in: H. de Swart (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98, Lecture Notes in Artificial Intelligence, vol. 1397, Springer-Verlag, Berlin, 1998, pp. 307–312, URL <download/1998/t98-paper.ps.gz>.
[4] V. Haarslev, R. Moller, Description of the RACER system and its applications, in: Proceedings of the Description Logics Workshop DL-2001, Stanford, CA, 2001, pp. 132–142.
[5] J. Broekstra, A. Kampman, F. van Harmelen, Sesame: a generic architecture for storing and querying RDF and RDF schema, in: The Semantic Web – ISWC 2002, Lecture Notes in Computer Science, vol. 2342, Springer, 2002, pp. 54–68.
[6] W. Farmer, J. Guttman, F. Thayer, Little theories, in: D. Kapur (Ed.), Proceedings of the Eleventh International Conference on Automated Deduction, Lecture Notes in Computer Science, vol. 607, Springer-Verlag, 1992, pp. 567–581.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

[7] P. Clark, J. Thompson, K. Barker, B. Porter, V. Chaudhri, A. Rodriguez, J. Thomere, S. Mishra, Y. Gil, P. Hayes, T. Reichherzer, Knowledge entry as the graphical assembly of components, in: Proceedings of the 1st International Conference on Knowledge Capture (K-Cap'01), 2001.
[8] E. Amir, S. McIlraith, Partition-based logical reasoning, in: 7th International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), 2000.
[9] S. McIlraith, E. Amir, Theorem proving with structured theories, in: B. Nebel (Ed.), Proceedings of IJCAI'01, Morgan Kaufmann, San Mateo, 2001, pp. 624–634.
[10] S. Lauritzen, D. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, Journal of the Royal Statistical Society 50.
[11] A. Rector, Modularisation of domain ontologies implemented in description logics and related formalisms including OWL, in: Proceedings of the 16th International FLAIRS Conference, AAAI, 2003.
[12] M. Buchheit, F.D.W. Nutt, A. Schaerf, Terminological systems revisited: Terminology = schema + views, in: Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), 1994.
[13] F. Giunchiglia, C. Ghidini, Local models semantics, or contextual reasoning = locality + compatibility, in: Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Morgan Kaufmann, 1998, pp. 282–289.
[14] A. Borgida, L. Serafini, Distributed description logics: directed domain correspondences in federated information sources, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2002: CoopIS, Doa and ODBase, LNCS, vol. 2519, Springer-Verlag, 2002, pp. 36–53.
[15] D. Brickley, R. Guha, A. Layman, Resource description framework (RDF) schema specification, Working draft, W3C, August 1998. <http://www.w3c.org/TR/WD-rdf-schema>.
[16] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein, Web ontology language (owl) reference version 1.0, Working draft, W3C, November 2002. <http://www.w3.org/TR/owl-ref/>.
[17] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, H. Stuckenschmidt, Contextualizing ontologies, Journal on Web Semantics 1 (4).
[18] R. Volz, A. Mdche, D. Oberle, Towards a modularized semantic web, in: Proceedings of the ECAI'02 Workshop on Ontologies and Semantic Interoperability, 2002.
[19] R. Volz, D. Oberle, R. Studer, Views for light-weight web ontologies, in: Proceedings of the ACM Symposium on Applied Computing SAC 2003, 2003.
[20] B. Cuenca-Grau, B. Parsia, E. Sirin, A. Kalyanpur, Modularity and web ontologies, in: P. Doherty, J. Mylopoulos, C. Welty (Eds.), Proceedings of the International Conference on Knowledge Representation and Reasoning KR-06, 2006, pp. 198–209.
[21] J. Bao, D. Caragea, V. Honavar, On the semantics of linking and importing in modular ontologies, in: I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. Aroyo (Eds.), Proceedings of the International Semantic Web Conference, ISWC-06, Athens, GA, 2006, Lecture Notes in Computer Science, vol. 4273, pp. 72–86.
[22] A. Borgida, On the relative expressiveness of description logics and predicate logics, Artificial Intelligence 82 (1–2) (1996) 353–367.
[23] D. Tsarkov, A. Riazanov, S. Bechhofer, I. Horrocks, Using vampire to reason with owl, in: Proceedings of the International Semantic Web Conference, 2004, pp. 471–485.
[24] H. Stuckenschmidt, F. van Harmelen, Information sharing on the semantic Web, in: Advanced Information Processing, Springer-Verlag, Berlin, Heidelberg.
[25] J. Gonzalez-Castillo, D. Trastour, C. Bartolini, Description logics for matchmaking of services, Technical Report HPL-2001-265, HP Laboratories Bristol, 2001.
[26] L. Li, I. Horrocks, A software framework for matchmaking based on semantic web technology, International Journal of Electronic Commerce 8 (4) (2004) 39–60.
[27] S. Bechhofer, I. Horrocks, D. Turi, The owl instance store: System description, in: Proceedings CADE-20, Lecture Notes in Computer Science, Springer-Verlag, 2005.
[28] I. Horrocks, U. Sattler, S. Tobies, Reasoning with individuals for the description logic $\mathcal{SHIQ}$, in: Proceedings of the 17th International Conference on Automated Deduction (CADE-17), Lecture Notes in Computer Science, Springer-Verlag, 2000.
[29] A. Borgida, L. Serafini, Distributed description logics: assimilating information from peer sources, Journal of Data Semantics 1 (2003) 153–184.
[30] L. Serafini, A. Borgida, A. Tamilin, Aspects of distributed and modular ontology reasoning, in: Proceedings of the International Joint Conference on Artificial Intelligence – IJCAI-05, Edinburgh, Scotland, 2005.
[31] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook – Theory, Implementation and Applications, Cambridge University Press, 2003.
[32] H. Stuckenschmidt, M. Klein, Integrity and change in modular ontologies, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, 2003. URL <http://www.cs.vu.nl/mcaklein/papers/IJCAI03.pdf>.
[33] S. Tobies, Complexity results and practical algorithms for logics in knowledge representation, PhD, thesis, LuFG Theoretical Computer Science, RWTH-Aachen, 2001.
[34] L. Serafini, A. Tamilin, DRAGO: distributed reasoning architecture for the semantic web, in: Proceedings of the Second European Semantic Web Conference (ESWC'05), Springer-Verlag, 2005.
[35] J. Banerjee, W. Kim, H.-J. Kim, H.F. Korth, Semantics and implementation of schema evolution in object-oriented databases, SIGMOD Record (Proc. Conf. on Management of Data) 16 (3) (1987) 311–322, URL http://doi.acm.org/10.1145/38713.38748.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001

[36] M. Klein, Change management for distributed ontologies, Ph.D. thesis, Vrije Universiteit Amsterdam, August 2004. URL<http://www.cs.vu.nl/mcaklein/thesis/>.

[37] E. Franconi, F. Grandi, F. Mandreoli, A sematic approach to schema evolution and versioning in object-oriented databases, in: Proceeding of CL 2000, Lecture Notes in Artificial Intelligence, vol. 1861, Springer-Verlag, 2000, pp. 1048–1062.

[38] M. Klein, D. Fensel, A. Kiryakov, D. Ognyanov, Ontology versioning and change detection on the web, in: 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Sigüenza, Spain, 2002, LNCS, vol. 2473, p. 197ff. URL <http://www.cs.vu.nl/mcaklein/papers/EKAW02.pdf>.

[39] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, L. Schneider, Sweetening ontologies with DOLCE, in: 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Sigüenza, Spain, 2002, Lecture Notes in Computer Science, vol. 2473, p. 166ff. URL <http://link.springer.de/link/service/series/0558/bibs/2473/24730166.htm>.

[40] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, Ontology library (final), Deliverable D18, EU/IST Project WonderWeb, December 2003. URL <http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf>.

[41] R. Volz, D. Oberle, S. Staab, R. Studer, Ontolift prototype, Deliverable D11, EU/IST Project WonderWeb, 2002.

[42] L. Serafini, H. Stuckenschmidt, H. Wache, A formal investigation of mapping languages for terminological knowledge, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence – IJCAI05, Edingurgh, UK, 2005.

[43] C. Ghidini, L. Serafini, Distributed first order logic – revised semantics, Technical Report, ITC-irst, January 2005.

Please cite this article in press as: H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. (2007), doi:10.1016/j.datak.2007.02.001