Integrity and Change in Modular Ontologies

Heiner Stuckenschmidt and **Michel Klein**
Vrije Universiteit Amsterdam
de Boelelaan 1081a, 1081HV Amsterdam
{heiner, michel.klein}@cs.vu.nl

Abstract

The benefits of modular representations are well known from many areas of computer science. In this paper, we concentrate on the benefits of modular ontologies with respect to local containment of terminological reasoning. We define an architecture for modular ontologies that supports local reasoning by compiling implied subsumption relations. We further address the problem of guaranteeing the integrity of a modular ontology in the presence of local changes. We propose a strategy for analyzing changes and guiding the process of updating compiled information.

1 Introduction

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, reasoning systems and more recently storage and query systems.[1] Most of these tools, however, treat ontologies as monolithic 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 is aimed at improving the current situation with respect to the following aspects.

Distributed Systems: In highly distributed systems such as the semantic web, modularity naturally exists in terms of physical location. Providing interfaces and mechanisms for connecting these natural modules is a prerequisite for easy maintenance [Heflin and Hendler, 2000].

Large Ontologies: Modularization also helps to manage very large ontologies we find for example in medicine or biology. Here modularity helps to maintain and reuse parts of the ontology as smaller modules are easier to handle than the complete ontology [Rector, 2003].

Efficient Reasoning: A specific problem that occurs in the case of distributed and large models is the problem of efficient reasoning. The introduction of modules with local semantics and clear interfaces will help to develop efficient reasoning methods [McIlraith and Amir, 2001].

1.2 Requirements

There are a couple of 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 refers to the conceptualization as well as the specific logical language used for the interpretation of objects, concepts or relations.

Self-Containment: In order to facilitate the reuse of individual modules we have to make sure that modules are self-contained. In especially, the result of certain reasoning tasks such as subsumption or query answering within a single module should be possible without having to access other modules.

Integrity: Having self-contained ontology modules may lead to inconsistencies that arise from changes in other ontology modules. We have to provide mechanisms for checking whether relevant knowledge in other systems has changed and for updating our modules accordingly.

1.3 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 following sections.

[1] An extensive overview is provided in the OntoWeb deliverable, available at `http://www.ontoweb.org/download/deliverables/D13_v1-0.zip`.

View-Based Mappings: We adopt the approach of view-based information integration. In particular, ontology modules are connected by conjunctive queries. This way of connecting modules is more expressive than simple one-to-one mappings between concept names but less expressive than the logical language used to describe concepts. We decide to sacrifice a higher expressiveness for the sake of conceptual simplicity and desirable semantic properties such as independence of the ontology langauge used.

Compilation of Implied Knowledge: 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 using the result. During reasoning, these axioms replace the query thus enabling local reasoning.

Change Detection and Automatic Update: Once a query has been compiled, the correctness of reasoning can only be guaranteed as long as the concept hierarchy of the queried ontology module does not change. 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 of the concept hierarchy.

The rest of the work is organized as follows. In section 2 we provide a definition of ontology modules based on a minimal notion of ontologies that fixes important properties we will use later on while leaving as much freedom for specific implementations as possible. Section 3 introduces our approach to self-containment in terms of compiled knowledge. The remainder of the paper is devoted to the problem of detecting changes and preserving integrity amongst modules in a system.

2 Modular Ontologies

Before we start investigating the problem of change and integrity, we define the notion of modular ontology we will use as a basis for our technical results.

2.1 Modules and Queries

A number of languages for encoding ontologies on the Web have been proposed (see [Gomez-Perez and Corcho, 2002] for an overview). In order to get a general notion of ontological knowledge, we define the general structure of an ontological module and its instantiation independent of a concrete language.

Definition 1 (Ontology Module) *A module is a triple $M = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ where $\mathcal{C}$ is a set of concept definitions, $\mathcal{R}$ is a set of relation definitions and $\mathcal{O}$ is a set of object definitions. Further, we define the signature of a module $\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ to be a triple $\langle \mathcal{CN}, \mathcal{RN}, \mathcal{ON} \rangle$, where $\mathcal{CN}$ is the set of all names of concepts defined in $\mathcal{C}$, $\mathcal{RN}$ the set of all relation names in $\mathcal{R}$ and $\mathcal{ON}$ the set of all object names occurring in $\mathcal{O}$.*

Queries over ontological knowledge are defined as conjunctive queries, where the conjuncts are predicates that correspond to concepts and relations of an ontology. Further, variables in a query may only be instantiated objects in that ontology.

Definition 2 (Ontology-Based Queries) *Let V be a set of variables disjoint from $\mathcal{ON}$ then an ontology-based query Q over a module $M = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ is an expressions of the form $q_1 \wedge \cdots \wedge q_m$ where q_i are query terms of the form $C(x)$ or $R(x, y)$ such that $x, y \in V \cup \mathcal{ON}$, $c \in \mathcal{CN}$ and $r \in \mathcal{RN}$ or are of the form $x = o$ where $x \in V$ and $o \in \mathcal{ON}$[2].*

The fact that all conjuncts relate to elements of the ontology allows us to determine the answer to ontology-based queries in terms of instantiations of the query that are logical consequences of the knowledge base.

2.2 Internal and External Definitions

The notion of module and query given above is a quite standard ones. What makes up a modular ontology now, is the possibility to use ontology-based queries in order to define concepts in one module in terms of a query over another module. For this purpose, we divide the set of concepts in a module into internally defined concepts $\mathcal{C}_I$ and externally defined concepts $\mathcal{C}_E$ resulting into the following definition of $\mathcal{C}$:

$$\mathcal{C} = \mathcal{C}_I \cup \mathcal{C}_E,\ \mathcal{C}_I \cap \mathcal{C}_E = \emptyset \tag{1}$$

Internally defined concepts are specified by using concept expressions in the spirit of description logics [Baader *et al.*, 2003]. We do not require a particular logic to be used.

Definition 3 (Internal Concept Definition) *An internal concept definition is an axiom of one of the following forms $C \sqsubseteq D, C \equiv D$ where $C \in \mathcal{CN}$ and D is a concept expression of the form $f(t_1, \cdots, t_n)$ where the terms t_i are either concept names or concept expressions and f is an n-ary concept building operator.*

Besides this standard way of defining concepts, we consider externally defined concepts that are assumed to be equivalent to the result of a query posed to another module in the modular ontology. This way of connecting modules is very much in spirit of view-based information integration which is standard technique in the area of database systems [Halevy, 2001]. The choice of conjunctive queries for connecting different modules is motivated by the trade-off between expressiveness of the mapping and conceptual as well as computational simplicity. Our approach is more expressive than simple one-to-one mappings; having more complex mappings would contradict the principle of loose coupling of different modules.

Definition 4 (External Concept Definition) *An external concept definition is an axiom of the form: $C \equiv M : Q$ Where M is a module and Q is an ontology-based query over the signature of M.*

A modular ontology is now simply defined as a set of modules that are connected by external concept definitions. In particular we require that all external definitions are contained in the modular system.

[2]Note that this may include data-type expressions as the type itself is can be considered to be a concept, the actual value a member of that concept and the comparison operator a special relation.

Definition 5 (Modular Ontology) *A modular ontology $\mathcal{M} = \{M_1, \cdots, M_m\}$ is a set of modules such that for each externally defined concept $C \equiv M_i : Q$, M_i is also a member of $\mathcal{M}$.*

We will use this notion of a modular ontology in the following to investigate the problem of integrity of logical reasoning across modules.

2.3 Semantics and Logical Consequence

We define a model-based semantics for modular ontologies using the notion of a distributed interpretation proposed by [Borgida and Serafini, 2002] in the context of distributed description logics:

Definition 6 (Distributed Interpretation) *A distributed interpretation $\Im = \langle \{\Im_i\}_{i \in Index}, r \rangle$ of a modular ontology $\mathcal{M}$ consists of interpretations $\Im_i$ for the individual module M_i over domains Δ_i, such that:*

- $C_i^{\Im} \subseteq \Delta_i$ *for all concept definitions* $C \in \mathcal{C}_i$
- $R_i^{\Im} \subseteq \Delta_i \times \Delta_i$ *for all relation definition* $R \in \mathcal{R}_i$
- $O_i^{\Im} \in \Delta_i$ *for all object definitions* $O \in \mathcal{O}_i$

and a function r associating to each pair of indices i, j a binary relation $r_{ij} \subseteq \Delta_i \times \Delta_j$. $r_{ij}(d)$ denotes the set $\{d' \in \Delta_j | (d, d') \in r_{ij}\}$; for every $D \subseteq \Delta_i$ $r_{ij}(D)$ denotes $\bigcup_{d \in D} r_{ij}(d)$.

The assumption of disjoint interpretation domains again reflects the principle of loose coupling underlying our approach. Based on the notion of a distributed interpretation we can define a model of a modular ontology as an interpretation that satisfies the constraints imposed by internal and external concept definitions. In contrast to [Borgida and Serafini, 2002], we do not introduce special operators for defining the relations between different domains, we rather interpret external concept definitions as constraints on the relation between the domains:

Definition 7 (Logical Consequence) *A distributed interpretation $\Im$ is a model for a modular ontology $\mathcal{M}$ if for every module M_i we have $\Im \models C$ for every concept definition C in M_i where $\models$ is defined as follows.*

- $\Im \models C \sqsubseteq D$, *iff* $C_i^{\Im} \subseteq D_i^{\Im}$
- $\Im \models C \equiv D$, *iff* $C_i^{\Im} = D_i^{\Im}$
- $\Im \models C \equiv M_j : Q$, *iff* $C^{\Im_i} = r_{ji}(Q^{\Im_j})$.

Here $Q^{\Im_j}$ denotes the interpretation of the set of answers to query Q. An axiom A logically follows from a set of axioms $\mathcal{S}$ if $\Im \models \mathcal{S}$ implies $\Im \models A$ for every model $\Im$. We denote this fact by $\mathcal{S} \models A$.

The actual definitions of concepts impose further constraints on the interpretation of a modular ontology. For the case of internally defined concepts, these constraints are provided by the definition of concept building operators of description logics. For the case of externally defined concepts, the situation is more complicated and will be discussed in more details in the next section.

3 Compilation and Local Reasoning

Using the notion of logical consequence defined above, we now turn our attention to the issue of reasoning in modular ontologies. For the sake of simplicity, we only consider the interaction between two modules in order to clarify the basic principles. Further, we assume that only one of the two modules contains externally defined concepts in terms of queries to the other module.

3.1 Implied Subsumption

As mentioned in the introduction, we are interested in the possibility of performing local reasoning. For the case of ontological reasoning, we focus on the task of deriving implied subsumption relations between concepts within a single module. For the case of internally defined concepts this can be done using well established reasoning methods [Donini *et al.*, 1996]. Externally defined concepts, however, cause problems: being defined in terms of a query to the other module, a local reasoning procedure will often fail to recognize an implied subsumption relation between these concepts. Consequently, subsumption between externally defined concepts requires reasoning in the external module as the following theorem shows.

Theorem 1 (Implied Subsumption) *Let E_1 and E_2 be two concepts in module M_i that are externally defined in module M_j by queries Q_1 and Q_2, then $\Im \models E_1 \sqsubseteq E_2$ if $\Im_j \models Q_1 \sqsubseteq Q_2$.*

Proof 1 $[\Im_j \models Q_1 \sqsubseteq Q_2] \Rightarrow [Q_1^{\Im_j} \subseteq Q_1^{\Im_j}] \Rightarrow [r_{ji}(Q_1^{\Im_j}) \subseteq r_{ji}(Q_1^{\Im_j})] \Rightarrow [E_1^{\Im_i} \subseteq E_1^{\Im_i}] \Rightarrow [\Im \models E_1 \sqsubseteq E_2]$

The result presented above implies the necessity to decide subsumption between conjunctive queries in order to identify implied subsumption relations between externally defined concepts. In order to decide subsumption between queries, we translate them into internally defined concepts in the module they refer to. A corresponding sound and complete translation is described in [Horrocks and Tessaris, 2000]. Using the resulting concept definition, to which we refer as *query concepts*, we can decide subsumption between externally defined concepts by local reasoning in the external ontology.

3.2 Compilation and Integrity

We can avoid the need to perform reasoning in external modules each time we perform reasoning in a local module using the idea of knowledge compilation [Cadoli and Donini, 1997]. The idea of compilation is to perform the external reasoning once and add the derived subsumption relations as axioms to the local module. These new axioms can then be used for reasoning instead of the external definitions of concepts. This set of additional axioms can be computed using Algorithm 1.

If we want to use the compiled axioms instead of external definitions, we have to make sure that this will not invalidate the correctness of reasoning results. We call this situation, where the compiled results are correct as integrity. We formally define integrity as follows:

Algorithm 1 Compile

Require: the module $M = \langle C_I \cup C_E, R, O \rangle$
Require: the external module $M_j = \langle C_j, R_j, O_j \rangle$
  for all $E \equiv M_j : Q \in C_E$ **do**
    $C'_E := C'_E \cup \{E \sqsubseteq C | C \in C_j, \Im_j \models E \sqsubseteq Q\}$
  end for
  return C'_E

Definition 8 (Integrity) *We consider integrity of two ontology modules M, M_j to be present if $M, M_j \models M^c$ where M^c is the result of replacing the set of external concept definitions in M by $compile(M, M_j)$.*

At the time of applying the compilation this is guaranteed by theorem 1, however, integrity cannot be guaranteed over the complete life-cycle of the modular ontology. The problem is, that changes to the external ontology module can invalidate the compiled subsumption relationships. In this case, we have to perform an update of the compiled knowledge.

4 Change Robustness

In principle, testing integrity might be very costly as it requires reasoning within the external ontology. In order to avoid this, we propose a heuristic change detection procedure that analyzes changes with respect to their impact on compiled subsumption relations. Work on determining the impact of changes on a whole ontology is reported in [Heflin and Hendler, 2000]. As our goal is to determine whether changes in the external ontology invalidates compiled knowledge, we have to analyze the actual impact of changes on individual concept definitions. We want to classify these changes as either *harmless* or *harmful* with respect to compiled knowledge.

4.1 Determining Harmless Changes

As compiled knowledge reflects subsumption relations between 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 query concepts. We first look at the effect of a set of modifications on individual concepts:

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 relate to the new version C':

1. the meaning of 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 concept becomes more general: $C \sqsubseteq C'$
3. the meaning of a concept is changed in such a way that 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'.

The same observations can be made for a relation before and after a change, denoted as R and R' respectively. The next question is how these different types of changes influences the interpretation of query concepts. 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 $c(Q)$ be the set of all concept names and $r(Q)$ the set of all relation names occurring in query 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 $C \sqsubseteq C' \Longrightarrow Q \sqsubseteq Q'$ and $C' \sqsubseteq C \Longrightarrow Q' \sqsubseteq Q$ where Q' is the query as being interpreted after changing C. Analogously, a change of R has the same effect on the complete query.*

Proof 2 (Sketch) *The idea of the proof is the following: Queries contain conjuncts of the form $C(x)$ or $R(x, y)$. Conjuncts of the first form are interpreted as $\{x | x \in C^{\Im}\}$. It directly follows that changing the interpretation of the concept C referred to in a conjunct of this type leads to the same change on the interpretation of the conjunct and because conjunction is interpreted as set intersection the whole query. Conjuncts of the second type are interpreted as $\{x | \exists y : (x, y) \in R^{\Im}\}$. The variable y can be further constraint by a conjunct of the first type. Again changes in the interpretation of the concept that further restricts y have the same effect on possible interpretations of y and therefore also on the interpretation of conjuncts of the second type. Using the same argument, we see that making R more general/specific (allowing more/less tuples in the relation) makes conjuncts of the second form more general/specific. Using these basic conclusions, we can proof the lemma by induction over the lengths of the path in the dependency graph of the query where nodes represent conjuncts and arcs co-occurrence of variables.*

We can exploit this relation between the interpretation of concepts and queries 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 two query concepts does not change if the more general(specific) query becomes even more general(specific) or stay the same. Combining these two observations, we derive the following characterization of harmless change.

Theorem 2 (Harmless Change) *A change is harmless with respect to compiled knowledge (i.e. $Q_1 \sqsubseteq Q_2 \Longrightarrow Q'_1 \sqsubseteq Q'_2$) if for all compiled subsumption relations $C_1 \sqsubseteq C_2$ where C_i is defined by query Q_i we have:*

- *$X' \sqsubseteq X$ for all $X \in c(Q_1) \cup r(Q_1)$*
- *$X \sqsubseteq X'$ for all $X \in c(Q_2) \cup r(Q_2)$*

Proof 3 *We assume that $X' \sqsubseteq X$ for all $X \in c(Q_1) \cup r(Q_1)$. Applying lemma 1 with respect to all $X \in c(Q_1) \cup r(Q_1)$ we*

derive $Q'_1 \sqsubseteq Q_1$. We further assume that $X \sqsubseteq X'$ for all $X \in c(Q_2) \cup r(Q_2)$. Using lemma 1 we get $Q_2 \sqsubseteq Q'_2$. This leads us to $Q'_1 \sqsubseteq Q_1 \sqsubseteq Q_2 \sqsubseteq Q'_2$. Theorem 2 is established by transitivity of the subsumption relation.

The theorem provides us with a correct but incomplete method for deciding whether a change is harmless. This basic method can be refined by analyzing the overlap of $c(Q_1)$ and $c(Q_2)$ in combination with the relations they restrict. This more accurate method is not topic of this paper, but it relies on the same idea as the theorem given above.

4.2 Characterizing Changes

Now 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 concepts 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 modification in the ontology. To to so, we have listed all possible change operations to an ontology according to the OWL-lite[3] knowledge model in the same style as done in [Banerjee *et al.*, 1987]. The list of operations is extendable to other knowledge models; we have chosen the OWL-lite model because of its simplicity and its expected important role on the Semantic Web. Apart from *atomic change operations* to an ontology — like add range restriction or delete subclass relation — the list also contains some *complex change operations*, which consist of multiple atomic operations and/or incorporate some additional knowledge. The complex changes are often more useful to specify effects than the basic changes. For example, for operations like concept moved up, or domain enlarged, we can specify the effect more accurately than for the atomic operations subclass relation changed and domain modified[4]. Atomic changes can be detected without using the knowledge in the ontology itself, only using the knowledge of the knowledge model, i.e. the language. These changes are detected at a structural level. To identify complex changes, we also need to use the content of the ontology itself. We are currently working on rules and heuristics to distill complex changes from sets of atomic changes [Klein and Noy, 2003]. Table 1 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 still growing as new complex changes are defined. A snapshot of the change ontology can be found online.[5] The specification of effects is not complete, in the sense that it describes "worst case" scenario's, and that for some operations the effect is "unknown" (i.e. unpredictable). In contrast to [Franconi *et al.*, 2000] who provides complete semantics of changes we prefer to use heuristics in order to avoid expensive reasoning about the impact of changes.

[3]See `http://www.w3.org/TR/owl-features/`.

[4]For a complete list, see `http://wonderweb.man.ac.uk/deliverables/D20.shtml`.

[5]`http://ontoview.org/changes/1/3/`

Operation	Effect
Attach a relation to concept C	C: Specialized
Complex: Change the superclass of concept C to a concept lower in the hierarchy	C: Specialized
Complex: Restrict the range of a relation R *(effect on all C that have a restriction on R)*	R: Specialized, C: Specialized
Remove a superclass 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) subclass A of C	C: No effect
Define a relation R as functional	R: Specialized

Table 1: Some modification to an ontology and their effects on the classification of concepts in the hierarchy.

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 other modules is still valid when the external ontology is changed. The complete procedure is as follows:

1. create a list of concepts and relations that are part of the "subsuming" query of any compiled axiom;
2. create another list of concepts and relations that are part of the "subsumed" query of any compiled axiom;
3. achieve the modifications that are performed in the external ontology;
4. use the modifications to determine the effect on the interpretation of the concept and relations.
5. check whether there are concepts or relations in the first, "subsuming", list that became more specific, or concepts or relations in the second, "subsumed", list that became more general, or concepts or relations in any of the lists with an unknown effect; if not, the integrity of the mapping is preserved.

Algorithm 2 Update

Require: Ontology Module M
Require: Ontology Module M_j
  for all compiled axioms $C_1 \sqsubseteq C_2$ in M^c **do**
    for all $X \in c(Q_1) \cup r(Q_1)$ **do**
      if effect on C is 'generalized' or 'unknown' **then**
        $M^c := Compile(M, M_j)$
      end if
    end for
    for all $X \in c(Q_2) \cup r(Q_2)$ **do**
      if effect on X is 'specialized' or 'unknown' **then**
        $M^c := Compile(M, M_j)$
      end if
    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 require in order to resume integrity. Otherwise

no action is taken, because the previously compiled knowledge is still valid. All the steps can be automated. A tool that helps to automate step 3 and 4 is described in [Klein *et al.*, 2002]. 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. It will also be able to detect some of the *complex* operations. The tool will also annotate the definitions in an ontology with the effect that the change has on its place in the hierarchy.

5 Conclusions

There is a growing need for applying the principle of modularity to representations of ontological knowledge in order to facilitate the creation, maintenance and re-use of knowledge. This paper contributes to the development of a theory of modular ontologies, focussing on the issue of reasoning in modular ontologies that change over time. The contributions of this paper is three-fold:

1. We propose an architecture for modular ontologies and analyze the role of mappings in logical reasoning across modules.
2. We describe a knowledge compilation approach that makes local reasoning within modules possible and define the notion of integrity.
3. We develop an update strategy that preserves integrity by identifying changes in ontology modules and deciding whether the compiled knowledge has to be updated or not.

We think that the approach described meets the practical needs of creating and using ontologies without missing a formal underpinning. It uses well-established representations of ontological knowledge and a rather simple and intuitive representation of mappings. Further, all of the supporting methods described can be automated in order to assist ontology engineers and developers of ontology-based systems. We deliberately chose to make some simplifications in order to be able to develop concise methods. First of all, these simplifications concern the restriction to a system of only two modules and the use of a rather weak heuristic for determining the effect of changes on compiled knowledge. In future work, we will investigate the impact of these simplifications and try to develop a more complete theory of the interaction in complex systems of modules and of the impact of changes on derived knowledge. Further one can as well imagine an external definition of relations using conjunctive queries with more than one free variable and reducing implied subsumption to the general problem of query containment under constraints [Calvanese *et al.*, 1998].

References

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[Banerjee *et al.*, 1987] Jay Banerjee, Won Kim, Hyoung-Joo Kim, and Henry F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *SIGMOD Record (Proc. Conf. on Management of Data)*, 16(3):311–322, May 1987.

[Borgida and Serafini, 2002] A. Borgida and L. Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In *Proceedings of the Intenational Conference on Cooperative Information Systems*, 2002.

[Cadoli and Donini, 1997] M. Cadoli and F.M. Donini. A survey on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997.

[Calvanese *et al.*, 1998] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[Donini *et al.*, 1996] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, pages 193–238. CSLI Publications, 1996.

[Franconi *et al.*, 2000] Enrico Franconi, Fabio Grandi, and Federica Mandreoli. A semantic approach for schema evolution and versioning in object-oriented databases. In *Computational Logic 2000*, number 1861 in Lecture Notes in Computer Science, pages 1048 – 1062, 2000.

[Gomez-Perez and Corcho, 2002] A. Gomez-Perez and O. Corcho. Ontology langauges for the semantic web. *IEEE Intelligent Systems*, January/February:54–60, 2002.

[Halevy, 2001] A.Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[Heflin and Hendler, 2000] Jeff Heflin and James Hendler. Dynamic ontologies on the web. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 443–449. AAAI/MIT Press, Menlo Park, CA, 2000.

[Horrocks and Tessaris, 2000] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.

[Klein and Noy, 2003] Michel Klein and Natalya F. Noy. A component-based framework for ontology evolution. Technical Report IR-504, Department of Computer Science, Vrije Universiteit Amsterdam, March 2003.

[Klein *et al.*, 2002] Michel Klein, Atanas Kiryakov, Damyan Ognyanov, and Dieter Fensel. Ontology versioning and change detection on the web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, Sigüenza, Spain, October 1–4, 2002.

[McIlraith and Amir, 2001] S. McIlraith and E. Amir. Theorem proving with structured theories. In B. Nebel, editor, *Proceedings of IJCAI'01*, pages 624–634, San Mateo, August 2001. Morgan Kaufmann.

[Rector, 2003] 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.