Modularization of Ontologies

WonderWeb:
Ontology Infrastructure for the Semantic Web

Heiner Stuckenschmidt
with contributions from: Michel Klein
Vrije Universiteit Amsterdam
email: michel.klein@cs.vu.nl

Identifier	Del 21
Class	Deliverable
Version	1.0
Date	26.6.2003
Status	Draft
Distribution	Public
Lead Partner	VUA

WonderWeb Project

This document forms part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2001-33052.

For further information about WonderWeb, please contact the project co-ordinator:

Ian Horrocks
The Victoria University of Manchester
Department of Computer Science
Kilburn Building
Oxford Road
Manchester M13 9PL
Tel: +44 161 275 6154
Fax: +44 161 275 6236
Email: wonderweb-info@lists.man.ac.uk

Contents

1 Motivation

Currently, research in the area of the semantic web is in a state where ontologies are ready to 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 [4, 30], reasoning systems [24, 21] and more recently storage and query systems (e.g. [7]). Most of these tools, however, tread ontologies as monolithic entities and provide little support for specifying, storing and accessing ontologies in a modular manner. Existing proposals trying to fill this gap lack a formal underpinning.

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 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 to 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 and clearly specified interfaces can help to overcome this problem.

Large Ontologies: Modularization is not only desirable in distributed environments, it also helps to manage very large ontologies we find for example in medicine or biology. These ontologies that 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 reuse as 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 that occurs in the case of distributed ontologies as well as very large models is the problem of efficient reasoning. While the pure size of the ontologies causes problems in the latter case, in a distributed setting, 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 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 or the interpretation of objects, classes or relations. Our architecture has to reflect this by providing an extremely loose coupling of modules. In especially, 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 especially, the result of certain reasoning tasks such as subsumption or query answering within a single module has to be possible 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 being independent from accessing 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. 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 are 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 knowledge engineering as well and database information systems.

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 [16]. They show the advantages of this modular approach in terms of reusability and reduced modelling 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 (see e.g. [11]). 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 [1] and first-order logic [29]. The work reported 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 (see e.g. [27]).

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 [31]. 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 reuse 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 [8]. They show that this distinction an 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 [19]. 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 [5].

The problem of combining and reasoning with ontological modules is has become of central importance in research on knowledge representation and reasoning on the so-called semantic web. Current proposals for languages to encode ontological knowledge on the world wide web, i.e. the RDF schema [6] and the web ontology language OWL [12] provide some basic mechanisms for combining modular representations. The abilities for combining 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. Volz and colleagues discuss different interpretations of the import statement that rage from purely syntactic to schema-aware interpretations of the imported knowledge [32]. An alternative way of relating different RDF models to each others that is much closer to our ideas is discussed

by Oberle [34] who defines a view language for RDF and defines some consistency constraints for the resulting model.

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 following sections.

View-Based Mappings: The first design decision made concerns the way, different ontology modules are connected. In our work, we adopt the approach of view-based information integration. In especially, ontology modules are connected by conjunctive queries. In especially, 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 such as directedness of the mapping. In especially, the definition of a query mapping does not influence the interpretation of the queried ontology.

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 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 using the result. During reasoning, 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 concept expression that contain all the information necessary to perform local reasoning. In our case this expression is the conjunction of all concepts of the other ontology module that subsume the query expression.

Change Detection and Automatic Update: Our approach of compiling mappings and adding the result to the ontology models 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 class hierarchy of the queried ontology module does not change. On the other hand, not every change in the hierarchy does really influence the compiled result. Problems only arise if concepts used in the query change or if the set of classes 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 of the class hierarchy in combination with the position of the affected class in that hierarchy. We further exploit an explicit representation of the dependencies between ontology modules in order to propagate changes in the system when necessary.

2 Modular Ontologies

In order to put a higher level modularization infrastructure for ontologies into place, extensions of existing technologies are necessary at different levels. On the syntactic level, we have to extend existing language standards like OWL with a language for defining module interfaces and mappings between different modules. On the semantic level, we have to define the interpretation of mappings as well as the relation between definitions in different modules in such a way that we achieve independence between modules. In this section we will present a framework for representing modular ontologies. In section 2.1 we define a syntax for representing modular ontologies and provide an intuitive description of its meaning. Section 2.2 underpins the syntax with a model-theoretic semantics for modular ontologies that uses the notion of a distributed interpretation across different abstract domains to define an novel notion of logical consequence that better fits the intuition of distributed models than the standard notion used by languages like OWL.

2.1 Syntax and Architecture

Before we turn our attention to modularized ontologies, we first clarify our view on ontologies by giving some basic definitions of models of ontological knowledge and their semantics.

2.1.1 Ontological Knowledge

A number of languages for encoding ontologies on the Web have been proposed (see [20] for an overview). In order to get a general notion of ontological knowledge, we define the general structure of a terminological knowledge base (ontology) and its instantiation independent of a concrete language.

Definition 1 (Terminological Knowledge Base) *A Terminological Knowledge Base $\mathcal{T}$ is a triple*

$$\mathcal{T} = \langle \mathcal{C}, \mathcal{R}, O \rangle$$

where $\mathcal{C}$ is a set of class definitions, $\mathcal{R}$ is a set of relation definitions and O is a set of object definitions.

Terminological knowledge usually groups objects of the world that have certain properties in common (e.g. cities or countries). A description of the shared properties is called

a class definition. Concepts can be arranged into a subclass-superclass relation in order to be able to further discriminate objects into sub-groups (e.g. capitals or European countries). Classes can be defined in two ways, by enumeration of its members or by stating that it is a refinement of a complex logical expressions. The specific logical operators to express such logical definitions can vary between ontology languages; the general definitions we give here abstract from these specific operators.

Definition 2 (Class Definitions) *A class definition is an axiom of one of the following forms:*

- $c \equiv (o_1, \cdots, o_n)$ *where c is a class definition and* $o_1, \cdots, o_n$ *are object definitions.*
- $c_1 \sqsubseteq c_2$ *where* c_1 *and* c_2 *are class definitions.*

Further, there is the universal class denoted as $\top$.

Objects of the same type normally occur in similar situations where they have a certain relation to each other (cities lie in countries, countries have a capital). These typical relations can often be specified in order to establish structures between classes. Terminological knowledge considers binary relations that can either be defined by restricting their domain and range or by declaring it to be a sub-relation of an existing one.

Definition 3 (Relation Definitions) *A relation definition is an axiom of one of the following forms:*

- $r \sqsubseteq (c_1, c_2)$ *where r is a role definition and* c_1 *and* c_2 *are class definitions.*
- $r_1 \sqsubseteq r_2$ *where* r_1 *and* r_2 *are role definitions.*

The universal role is defined as $\top \times \top$.

Sometimes single objects (e.g. the continent Europe) play a prominent role in a domain of interest, or the membership of a concept is defined by the relation to a specific object (European countries are those contained in Europe). For this purpose ontology languages often allow to specify single objects, also called instances. In our view on terminological knowledge, instances can be defined by stating their membership in a class. Further, we can define instances of binary relations by stating that two objects form such a pair.

Definition 4 (Object Definitions) *An object definition is an axiom of one of the following forms:*

- $o : c$ *where c is a class definition and o is an individual.*
- $(o_1, o_2) : r$ *where r is a relation definition and* o_1, o_2 *are object definitions.*

In the following, we will consider terminological knowledge bases that consist of such axioms. Of course, any specific ontology language will have to further instantiate these definitions to specify logical operators between classes etc, but for the purposes of this paper, these general definitions are sufficient. Further, we define the signature of a terminological knowledge base to be a triple $\langle \mathcal{CN}, \mathcal{RN}, \mathcal{IN} \rangle$, where $\mathcal{CN}$ is the set of all names of classes defined in $\mathcal{C}$, $\mathcal{RN}$ the set of all relation names and $\mathcal{IN}$ the set of all object names occurring in the knowledge base.

2.1.2 Internal and External Definitions

The notion of ontology 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 [2]. We do not require a particular logic to be used.

Definition 5 (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.

Definition 6 (Terminological Queries) *Let V be a set of variables disjoint from $I\mathcal{N}$ then an terminological query Q over a knowledge base $\mathcal{T}$ is an expressions of the form*

$$q_{1_i} \wedge \cdots \wedge q_{m_i}$$

where q_i are query terms of the form $x : c$ or $(x,y) : r$ such that $x,y \in V \cup I\mathcal{N}$, $C \in \mathcal{CN}$ and $R \in \mathcal{RN}$.

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 [22]. 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. We now use the notion of ontology based query in order to define concepts using queries over a different ontology (module) that have exactly one free variable.

Definition 7 (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 with exactly one free variable.*

Further, we allow relations to be defined in terms of query expressions with two free variables. By convention, we call these variables x and y where x always denotes the variable in the first and y the variable in the second place of the binary relation. Analogously to external concept definitions, we get the following definition for externally defined relations $R_{\mathcal{E}}$

Definition 8 (External Relation Definition) *An external relation definition is an axiom of the form: $R \equiv M : Q$ Where M is a module and Q is an ontology-based query over the signature of M with exactly two free variable.*

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

Definition 9 (Modular Ontology) *A modular ontology $\mathcal{M} = \{M_1, \cdots, M_m\}$ is a set of modules such that for each externally defined concept $C \equiv M_i : Q$ and each external relation defintion $R \equiv M_i : Q$ 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.

2.2 Semantics and Logical Consequence

After having defined a representation syntax for modular ontologies, we now have to define how a modular ontology should be interpreted. Such a semantic underpinning in necessary to define the notion of logical consequence which serves as a basis for any kind of reasoning. Further, having a formal semantics makes it easier to compare our model to existing proposals for ontologies on the web as well as to investigate the formal properties of the kind of mapping relations chosen.

When defining the semantics of our model, we have to find a trade-off between backward compatibility with existing standards and new ways of defining logical semantics that better fit the distributed nature of a modular ontology. In order to meet both requirements, we define a local semantics that applies to individual modules and a distributed semantics that defines how the relations between elements in different modules are interpreted. The local semantics directly corresponds to the Tarskian style semantics of description logics and is therefore very close to the semantics of OWL-DL which can be seen as a special kind of description logics. The distributed semantics borrows from the notion of distributed first order logics and more specifically distributed description logics defining the interaction between different local models referring to the local semantics.

2.2.1 Local Semantics

We can define semantics and logical consequence of a terminological knowledge base using an interpretation mapping $.^{\Im}$ into an abstract domain Δ such that:

- $c^{\Im} \subseteq \Delta$ for all class definitions c in the way defined above
- $r^{\Im} \subseteq \Delta \times \Delta$ for all relation definition r
- $o^{\Im} \in \Delta$ for all object definitions o

This type of denotational semantics is inspired by description logics [13], however, we are not specific about operators that can be used to build class definitions which are of central interest of these logics. Using the interpretation mapping, we can define the notion of a model in the following way:

Definition 10 (Model of a Terminological Knowledge Base) *An interpretation $\Im$ is a model for the knowledge base $\mathcal{T}$ if $\Im \models A$ for every axiom $A \in (\mathcal{C} \cup \mathcal{R} \cup O)$ where $\models$ is defined as follows.*

- $\Im \models c \equiv (o_1, \cdots, o_n)$, *iff* $c^\Im = \{o_1^\Im, \cdots, o_n^\Im\}$
- $\Im \models c_1 \sqsubseteq c_2$, *iff* $c_1^\Im \subseteq c_2^\Im$
- $\Im \models r \sqsubseteq (c_1, c_2)$, *iff* $r^\Im \subseteq c_1^\Im \times c_2^\Im$
- $\Im \models r_1 \sqsubseteq r_2$, *iff* $r_1^\Im \subseteq r_2^\Im$
- $\Im \models o : c$, *iff* $o^\Im \in c^\Im$
- $\Im \models (o_1, o_2) : r$, *iff* $(o_1^\Im, o_2^\Im) \in r^\Im$

These definitions enable us to perform reasoning using the notion of logical consequence:

Definition 11 (Logical Consequence) *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 fact that all conjuncts relate to elements of the ontology allows us to determine the answer to terminological queries in terms of instantiations of the query that are logical consequences of the knowledge base it refers to:

Definition 12 (Query Answers (Halevy 2001)) *The answer of a query Q containing variables $v_1, \cdots, v_k$ over a knowledge base T is a set of tuples $(i_1, \cdots, i_k)$ such that $T \models Q'$ where Q' is the query obtained from Q by substituting $v_1, \cdots, v_k$ by $i_1, \cdots, i_k$. The projections of the answer tuples to variables occurring in the head of a query Q is denoted as $res(Q)$ and referred to as the answer set of Q.*

The computation of query answers in the sense being defined above is the main inference task of peers within a system. In the next section we will discuss a logically well-founded approach for computing such answers.

As our goal is to ensure the integrity of distributed ontologies with respect to logical reasoning, the notion of logical consequence is a central one that will be used to establish our technical results. In especially, we use the notions of implied subsumption and membership:

Definition 13 (Subsumption and Membership) *Let $\langle \Delta, .^\Im \rangle$ be a structure with domain Δ and interpretation $.^\Im$ such that $C^\Im \subseteq \Delta$ for every $C \in \mathcal{CN}_O$, $R^\Im \subseteq \Delta \times \Delta$ for every $R \in \mathcal{RN}_O$ and $I^\Im \in \Delta$ for every $I \in I\mathcal{N}$.*

- *An instance a is said to be member of a concept C, denoted as $a : C$ iff $\mathcal{T} \models I^{\mathfrak{I}} \in C^{\mathfrak{I}}$*
- *A pair of instances (a,b) is said to be member of a Relation R, denoted as $(a,b) : R$ iff $\mathcal{T} \models (a^{\mathfrak{I}}, b^{\mathfrak{I}}) \in R^{\mathfrak{I}}$.*
- *A concept C is subsumed by another concept D, denoted as $C \sqsubseteq D$ iff $\mathcal{T} \models C^{\mathfrak{I}} \subseteq D^{\mathfrak{I}}$.*

Analogously, we can define the notion of subsumption between queries in terms of the subset relation between their result sets:

Definition 14 (Query Subsumption) *Let $\mathcal{T} = \langle \mathcal{C}, \mathcal{R}, O \rangle$ and Q_1, Q_2 conjunctive queries over $\mathcal{T}$. Q_1 is said to be subsumed be another query Q_2 denoted by $Q_1 \sqsubseteq Q_2$ if for all possible sets of object definitions of a terminological knowledge base the answers for Q_1 is a subset of the answers for Q_2 : $(\forall O : \{t | \mathcal{T} \models Q_1(t)\} \subseteq \{t | \mathcal{T} \models Q_2 t\}$*

2.2.2 Global Semantics

We define a model-based semantics for modular ontologies using the notion of a distributed interpretation proposed by [5] in the context of distributed description logics:

Definition 15 (Distributed Interpretation) *A distributed interpretation $\mathfrak{I} = \langle \{\mathfrak{I}_i\}_{i \in Index}, r \rangle$ of a modular ontology $\mathcal{M}$ consists of interpretations $\mathfrak{I}_i$ for the individual module M_i over domains Δ_i, such that:*

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

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

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 [5], 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 16 (Logical Consequence) *A distributed interpretation $\mathfrak{I}$ is a model for a modular ontology $\mathcal{M}$ if for every module M_i we have $\mathfrak{I} \models X$ for every concept or relation definition X in M_i where $\models$ is defined using definition 10 for internal definitions and the following equations:*

- *$\mathfrak{I} \models C \equiv M_j : Q$, iff $C^{\mathfrak{I}_i} = b_{ji}^1(Q^{\mathfrak{I}_j})$.*
- *$\mathfrak{I} \models R \equiv M_j : Q$, iff $R^{\mathfrak{I}_i} = b_{ji}^2(Q^{\mathfrak{I}_j})$.*

Here $Q^{\mathfrak{I}_j}$ denotes the interpretation of the set of answers to query Q. An axiom A logically follows from a set of axioms $\mathcal{S}$ if $\mathfrak{I} \models \mathcal{S}$ implies $\mathfrak{I} \models A$ for every model $\mathfrak{I}$. 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 Expressiveness of the Model

Different from the mainstream work on distributed ontology definitions, our approach uses a mapping language that is different from the logical language used to specify the local ontologies themselves. In particular, we use conjunctive queries over concepts and binary relations. At first glance this seems to be a serious restriction of the expressiveness of our language as mappings contain no negation, disjunction or other terminological operators. A careful investigation of the semantics of our model, however, reveals that the use of conjunctive queries actually leads to a higher expressiveness as opposed to the standard approach of linking ontologies by directly referring to elements of remote models in a local specification (compare [12]). In this section we show that the direct reference scheme used in languages like OWL can be simulated using a trivial mapping scheme, further we argue that our model is more expressive than the direct use of elements, because it allows to specify relations in terms of complex expressions. Finally, we sketch how our model can be extended in a straightforward way to also capture relations of arbitrary arity.

3.1 Resembling OWL Import

Aiming at the semantic web, the langauge we have to compare ourselves to is the Web Ontology Langauge OWL. In the current proposals for OWL, the notion of mapping is not explicitly contained in the language. The abilities for combining 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. While being quite simple, this way of connecting ontologies is quite flexible and allows for complex arrangements of elements from different models into one expression. In this section, we show that this ability can easily be resembled using our model using examples from the OWL language guide. The basic idea is the following: we create a local copy C of each external concept E involved using a trivial mapping of the form $C(X) \equiv M : E(X)$ and then combine these local copies in a complex definition using OWL class building operators.

3.1.1 Simple References

The most basic kind of reference to other ontologies mentioned in the OWL documentation is to state the equivalence of two classes using the `owl:equivalentClass` statement. The following example is taken from the OWL langauge guide:

```
<owl:Class rdf:ID="Wine">
  <owl:equivalentClass rdf:resource="&vin;Wine"/>
</owl:Class>
```

Assuming that the external ontology, described by the prefix `vin` is imported by the local ontology, this statements claims that the extension of the two concepts are actually the same. We can model this constraint on the interpretation of the local ontology using the following trivial definition of the external concept Wine:

$$Wine(X) \equiv M_{vin} : Wine(X)$$

In this case, we could directly encode the OWL reference mechanism in terms of our model. Having restricted the definitions of external concepts to equivalence statements, however, we cannot directly encode the weaker `subclassOf` relation to external concepts like the one in the example below:

```
<owl:Class rdf:ID="WineGrape">
  <rdfs:subClassOf rdf:resource="&food;Grape" />
</owl:Class>
```

At this point, we have to apply the modelling trick mentioned above: We create a local copy of the concept 'Grape' using the same trivial mapping as above and declare the local concept 'WineGrape' to be a subclass of this local copy:

$$\begin{aligned} C &\equiv M_{food} : Grape(X) \\ WineGrape(X) &\sqsubseteq C \end{aligned} \tag{2}$$

3.1.2 Combining Internal and External Definitions

The simple strategy of creating local copies of external concepts allows us to easily combine external and local definitions into more complex concepts. Again, we use an example from the OWL language guide, where Wine is defined as a subclass of the intersection of potable liquids and things made from grapes. Here, potable liquids are defined elsewhere, whereas the restricted relation 'madeOfGrapes' is contained in the local ontology:

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
```

```
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
      1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
```

Using the same strategy as before, we create a copy of the externally defined concept 'potatbleLiquid'. Using this copy, we simply define the concept Wine locally using the Description logic counterparts of the OWL operations used in the example:

$$\begin{aligned} C(X) &\equiv M_{food} : PotableLiquid(X) \\ Wine &\sqsubseteq C \sqcap (\leq 1\, madeFromGrape) \end{aligned} \tag{3}$$

3.1.3 Complex External References

The ability to make complex assertions about local copies from (maybe different) models also helps us to overcome the restricted expressiveness of our mapping language. Having restricted external definitions to conjunctive queries, we cannot directly express conjunctions or negation in the definitions. However, we can used disjunction, negation and other OWL operators locally in order to define complex concepts on the basis of local copies of concepts from other ontologies. In order to illustrate this possibility, we use the following concept definition that uses concepts from three different external models:

```
<owl:Class rdf:ID="LiquidPoison">
  <rdfs:subClassOf>
    <owl:Class rdf:resource="&physics;LiquidSubstance"/>
    <owl:Class>
        <owl:unionOf rdf:parseType="owl:collection">
            <owl:Class rdf:resource="&medicine;Drug"/>
            <owl:Class>
                <owl:completementOf rdf:resource="&food;PotableLiquid"/>
            </owl:Class>
        </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

In order to directly capture this definition in terms of an external concept definition, the mapping language would have to contain disjunction (for expressing `owl:unionOf` and negation (for expressing `owl:complementOf`). Instead, we resemble can the above concept expression in the following way that leads us to an internal concept expression with the same meaning as the example above.

$$C_1(X) \equiv M_{physics} : LiquidSubstance(X)$$

$$\begin{aligned} C_2(X) &\equiv M_{medicine} : Drug(X) \\ C_3(X) &\equiv M_{food} : PotableLiquid(X) \\ LiquidPoison &\sqsubseteq C_1 \sqcap (C_2 \sqcup \neg C_3) \end{aligned} \tag{4}$$

3.1.4 Relation Definitions

Concerning the definition of Relations, the abilities of OWL are quite limited. Most of the assertions that can be made about relations do not depend on other elements of the ontologies but solely address the mathematical properties of a relation (such as transitivity or functionality). The only assertions than can be made about a relations and its dependence on elements from other ontologies are `owl:inverseOf` as well as domain and range restrictions. A slightly modified example from the OWL documentation is the following:

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <owl:inverseOf rdf:resource="&wine;usedFor"/>
  <rdfs:domain rdf:resource="&wine;Wine"/>
  <rdfs:range rdf:resource="&wine;WineGrape"/>
</owl:ObjectProperty>
```

In order to capture this definition in our mapping framework, we can combine the direct use of our mapping language and the use of local copies. Defining a relation to be the inverse of an external one can directly be done using a mapping query and inverting the order of the return variables:

$$madeFromGrapes(X,Y) \equiv M_{wine} : usedFor(Y,X) \tag{5}$$

For the domain and range restrictions, we create local copies and define the relation to range over these local copies (compare section on axioms for defining ontological knowledge). As both kinds of restrictions, the mapping on the inverse of an external relation and the restriction local restriction of the domain and range apply to the definition of the 'madeFromGrape' relation, the semantics of its definition is the same as for the example definition.

$$\begin{aligned} C_1(X) &\equiv M_{wine} : Wine(X) \\ C_2(X) &\equiv M_{wine} : WineGrape(X) \\ madeFromGrapes &\sqsubseteq (C_1 \times C_2) \end{aligned} \tag{6}$$

3.2 Beyond OWL

The examples given above raise the questions for the advantages of a mapping language based on conjunctive queries. In this section, we argue that our mapping language extents the expressiveness of existing ontology languages that are solely based on description logics and in particular OWL by adding more possibilities for defining properties. In contrast to existing approaches for combining description logics with rule languages such

as [28] or [15], our approach only allows to use rules in a very specific way, namely to define relations between disjoint interpretation domains and does therefore not suffer from the technical problems of many other approaches. As a consequence, we can allow for complex terminological definitions such as the ones described above. In addition, we can define local relations by complex expressions over predicates in another model. Examples of such definitions that go beyond the expressiveness of OWL are given below.

3.2.1 Combining Relations

The major advance of our approach over the abilities of OWL is the possibility to intentionally define relations using concepts, relations and also instances of a remote model. Based on these definitions, our model allows to derive subsumption relations between externally defined relations (see definition 14) while OWL only allows to explicitly state subsumption relations between relations and use them to derive subsumption between relations. The example below describes the relation between employees and the companies they were employed by in a particular year:

$$\begin{aligned} employedIn2003(X,Y) \quad &\equiv \quad M : employmentContract(Z) \wedge employee(Z,X) \\ &\quad\ \wedge year(Z,2003) \wedge employer(Z,Y) \end{aligned} \tag{7}$$

If we now consider the more general relation of legal partners defined by the more general concepts of contract and beneficiary without reference to a particular year. Assuming that the model M provides the corresponding background knowledge $employmentContract \sqsubseteq contract$, $employee \sqsubseteq beneficiary$ and $employer \sqsubseteq beneficiary$ we can, based on the notion of logical consequence derive that the following relation subsumes the one described above:

$$legalpartner(X,Y) \equiv M : contract(Z) \wedge beneficiary(Z,X) \wedge beneficiary(Z,Y) \tag{8}$$

3.2.2 N-ary Relations

A more ambitions extension to OWL expressiveness, that is supported by our model (though it is not worked out in this paper) is the ability to express relations of an arbitrary arity. This is supported by the mapping language as well as the formal semantics of our approach. In our model, an n-ary relation can be defined using a query expression with n free variables. An example for a tertiary relation that connects an employee with his or her employer depending on a certain year is given below:

$$\begin{aligned} employed(X,P,Y) \quad &\equiv \quad M : employmentContract(Z) \wedge beneficiary(Z,X) \\ &\quad\ \wedge year(Z,P) \wedge employer(Z,Y) \end{aligned} \tag{9}$$

On the semantic level, relations of higher arity are supported by the use of the relations b_{ij}^k that connect the different interpretation domains. Up to now we are only using these

relations with the arity parameter k set to 1 for concepts and 2 for relations. The semanics of the tertiary relation above can be defined using the relation b^3_{ij}. Reasoning about these relations is equivalent to the problem of query containment under constraints which is known to be decidable for many interesting cases [10].

4 Reasoning in Modular Ontologies

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. 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 [14]. 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. We define the notion of implied subsumption starting with subsumption between atomic concepts before extending the results to arbitrarily complex concept expressions

4.1 Atomic Concepts and Relations

The most simple case of implied subsumption is the case where we want to decide whether two externally defined concepts subsume each other. Assuming that these concepts are solely defined in terms of their mapping to another ontology, we can define when these concepts subsume each other on the basis of query subsumption in the external ontology:

Theorem 1 (Implied Subsumption) *Let E_1 and E_2 be two concepts (or relations) 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 *For $a \in \{1,2\}$ he have:*

$$\begin{aligned} \Im_j \models Q_1 \sqsubseteq Q_2 &\Rightarrow Q_1^{\Im_j} \subseteq Q_1^{\Im_j} \\ &\Rightarrow b^a_{ji}(Q_1^{\Im_j}) \subseteq b^a_{ji}(Q_1^{\Im_j}) \\ &\Rightarrow E_1^{\Im_i} \subseteq E_1^{\Im_i} \\ &\Rightarrow \Im \models E_1 \sqsubseteq E_2 \end{aligned} \tag{10}$$

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 [25]. 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.

4.2 Preservation of Boolean Operators

Things become a bit more complicated when we consider the case where externally defined concepts are further used to define complex concepts. What is needed is a general result on the preservation of subsumption relationships between concept expressions in different modules that are defined in the same way. In the following we will call these expressions isomorphic.

Definition 17 (Isomorphic Concepts) *Let $i:C$ and $j:D$ be two concepts defined in modules M_i and M_j respectively, then $i:C$ and $i:D$ are said to be isomorphic if*

1. $i:C(x) = M_j:D(x)$ *or*
2. $i:C \equiv f(E_1,...,E_n)$, $j:D \equiv f(F_1,...,F_n)$, E_i *and* F_i *are isomorphic*

We formulate the following hypothesis about isomorphic concepts: For every pair of isomorphic concepts C and D we have

$$C^{\Im_i} = b^1_{ji}(D^{\Im}_i) \tag{11}$$

We try to prove the hypothesis by induction over the definition of isomorphic concepts. The induction hypothesis is directly established by definition 16. We therefore consider case 2 in definition 17. From the induction hypothesis, we know that $E_i^{\Im_i} = b^1_{ji}(F_j^{\Im_j})$. As

$$C^{\Im_i} = b^1_{ji}(D^{\Im}_i) \Leftarrow C^{\Im_i} = b^1_{ji}(f(F_1,....F_n)^{\Im_j})$$

in order to prove the lemma we have to show that b^1_{ji} distributes over f, in particular, that:

$$b^1_{ji}(f(F_1,...,F_n)^{\Im_j}) = f(b^1_{ji}(F_1^{\Im_j}),...,b^1_{ji}(F_n^{\Im_j}))^{\Im_i}$$

Because in this case, we can use the induction hypothesis to replace the arguments of f resulting in: $C^{\Im_i} = f(E_i^{\Im_j},...,E_n^{\Im_j})^{\Im_i}$ Which directly follows from the definition.

We investigate above statement with respect to the boolean operators over class names. For the sake of readability, we use b instead of b^1_{ji} to denote the semantic relation between M_j and M_i.

Disjunction Conjunction is defined in terms of the union of the extensions of concepts. We have to show that: $b(C^{\mathfrak{I}_j} \cup D^{\mathfrak{I}_j}) = b(C^{\mathfrak{I}_j}) \cup b(D^{\mathfrak{I}_j})$.

($\subseteq$) for each element $x \in b(C^{\mathfrak{I}} \cup D^{\mathfrak{I}})$ there is an element $y \in (C^{\mathfrak{I}} \cup D^{\mathfrak{I}})$ with $b(y,x)$. For this y we know that either $y \in C^{\mathfrak{I}}$ or $y \in C^{\mathfrak{I}}$. As b is defined for y, we also have an object x' with $b(y,x')$ and either $x' \in b(C^{\mathfrak{I}_j})$ or $x' = b(D^{\mathfrak{I}_j})$ and therefore $x' \in b(C^{\mathfrak{I}} \cup D^{\mathfrak{I}})$. What is left to be shown is that $x' = x$. This actually is only given id b is a function which we have to take as a premise.

($\supseteq$) For each element $x \in b(C^{\mathfrak{I}}) \cup b(C^{\mathfrak{I}})$ we know that $x \in b(C^{\mathfrak{I}})$ or $x \in b(D^{\mathfrak{I}})$. Therefore there is an element y with $b(y,x)$ and either $y \in C^{\mathfrak{I}}$ or $y \in D^{\mathfrak{I}}$. We conclude that $y \in (C^{\mathfrak{I}} \cup D^{\mathfrak{I}})$. As b is defined for y there is an element x' with $b(y,x')$ and $x' \in b(C^{\mathfrak{I}} \cup D^{\mathfrak{I}})$. As above what is left to be shown is $x = x'$ which is the case if b is a function.

Conjunction Conjunction is defined in terms of the intersection of the extensions of concepts. We have to show that: $b(C^{\mathfrak{I}_j} \cap D^{\mathfrak{I}_j}) = b(C^{\mathfrak{I}_j}) \cap b(D^{\mathfrak{I}_j})$.

($\subseteq$) For each element $x \in b(C^{\mathfrak{I}} \cap D^{\mathfrak{I}})$ there is an element $y \in (C^{\mathfrak{I}} \cap D^{\mathfrak{I}})$ with $b(y,x)$. For this y we know that $y \in C^{\mathfrak{I}}$ and $y \in D^{\mathfrak{I}}$. We conclude (as b is defined for y) that there is an x' with $b(y,x')$ and $x' \in b(C^{\mathfrak{I}}) \cap b(C^{\mathfrak{I}})$. What is left to be shown is that $x' = x$. This actually is only given id b is a function which we have to take as a premise.

($\supseteq$) For each element $x \in b(C^{\mathfrak{I}}) \cap b(C^{\mathfrak{I}})$ we know that $x \in b(C^{\mathfrak{I}})$ and $x \in b(D^{\mathfrak{I}})$. Therefore, there are elements y_1, y_2 with $y_1 \in C^{\mathfrak{I}}, b(y,x)$ and $y_2 \in D^{\mathfrak{I}}, b(y',x)$. What we are looking for is an element y with $y \in (C^{\mathfrak{I}} \cap D^{\mathfrak{I}}), b(y,x)$. In this case we could use the same argument as above to show that the subset equation holds if b is a function. Actually, he have such an element y if we could show that $y_1 = y_2$. This is actually the case if b is an injective function which is another premise.

Negation OWL uses negation in terms of the `owl:complementOf` operator. Its semantics is defined in terms of set complement with respect to the domain of interpretation, i.e. $(\neg C)^{\mathfrak{I}_i} = \Delta_i - C^{\mathfrak{I}_i}$. We have to show that $b(\Delta_j - C^{\mathfrak{I}}) = \Delta_i - b(C^{\mathfrak{I}})$.

($\subseteq$) For every $x \in b(\Delta_j - C^{\mathfrak{I}})$ we know that there is an element y with $b(y,x)$ and $y \notin C^{\mathfrak{I}}$. As we know that b is defined for y, there is an x' with $b(y,x')$. Two things need to shown for this x'. That it is not in $b(C^{\mathfrak{I}})$ and that $x = x'$. The latter is given if b is a function. The former can be guaranteed if b is injective since for each element $x'' \in b(C^{\mathfrak{I}})$ there is an element $y' \in C^{\mathfrak{I}}$. From the injectivity of b it would follow that $y' = y$ which results in a conflict as we know that $y \notin C^{\mathfrak{I}}$.

($\supseteq$) Let $x \in \Delta_i - b(C^{\mathfrak{I}})$. We assume that there is an element y with $b(y,x)$ and $y \notin C^{\mathfrak{I}}$. We assume that $y \in C^{\mathfrak{I}}$. Therefore there exists an element x' with $x' \in b(C^{\mathfrak{I}})$. For the case that b is functional, we can derive a conflict, because in this case $x = x'$ and $x \notin b(C^{\mathfrak{I}})$. If follows that $y \in \Delta_j - C^{\mathfrak{I}}$. As b is defined for y there is an element x'' with $b(y,x'')$ such hat $x'' \in b(\Delta_j - C^{\mathfrak{I}})$. Again, if b is functional, we have $x = x''$ which establishes the result

under the assumption that we can find a suitable y. This, however, can only be guaranteed if b is also surjective, as otherwise it might be the case that x is not in the image of b. Summarizing, we can say that the set inclusion only holds if b is a bijective function.

4.3 Intuition and Implications for Inter-Module Reasoning

We can use the results about the semantic relation between the extensions of isomorphic concepts we presented above in order to extend theorem 1. While the theorem only makes assertions about concepts that are directly defined by external mappings, we saw above that under certain assumptions there is also a semantic connection between concepts that are not directly connected but built from other connected concepts (see equation 11). The table below summarizes these results:

b^1_{ji}	Atomic Concepts	Disjunction	Conjunction	Negation
relation	X			
function	X	X		
injective function	X	X	X	
bijective function	X	X	X	X

We see that there is a semantic relation between isomorphic concepts defined using only disjunction is provided if the semantic mapping is functional. If conjunction is used as well, the semantic mapping has to be an injective function in order to guarantee that a semantic relation exists between isomorphic concepts. If negation is used to define concepts, only a semantic mapping which is a bijective function implies equation 11. Based on this observation, we formulate the following extension of theorem 1

Theorem 2 (Implied Subsumption (extended)) *Let E_1 and E_2 be two concepts in module M_i and $\Im_i \not\models E_1 \sqsubseteq E_2$. Let further be $F_2 and F_2$ be concepts in module M_j with $\Im_j \models F_1 \sqsubseteq F_2$. We have $\Im \models E_1 \sqsubseteq E_2$ if:*

1. *Theorem 1 applies.*
2. *E_1, F_1 and E_2, F_2 are isomorphic, b^1_{ji} is a function and only disjunction is used to define concepts*
3. *E_1, F_1 and E_2, F_2 are isomorphic, b^1_{ji} is an injective function and only disjunction and conjunction are used to define concepts.*
4. *E_1, F_1 and E_2, F_2 are isomorphic, b^1_{ji} is a bijective function and only disjunction, conjunction and negation are used to define concepts.*

Proof 2 (Sketch) *The theorem is proven in the same way as theorem 1 where the third step of the derivation is justified by equation 11 which has been shown in the last section.*

The crucial question connected to these technical results is about the suitability of the assumptions we make about the semantic relation. In order to get an idea about these assumptions, we will take a look at the formal properties of the semantic relation and

discuss the intuition connected with these properties.

In the most general case, b_{ji} is just a general relation with no further restrictions. As a result it provide a high flexibility with respect to the links that exist between modules. This general relation allows for example to connect models with different levels of granularity as one element in the domain one module may correspond to several elements of the domain of the other module. This flexibility leads to a very loose coupling of different modules as no operators are preserved. In principle, we only know connections between modules that are explicitly stated. This changes when we assume that the semantic relation is functional. In this case every element in the domain of M_i only corresponds to exactly one element in M_i. This means that the goal module is at least as fine grained (or exact) as the target. Still it can be the case that the target is an abstraction, because more than one element of the goal domain correspond to one element in the target domain. Choosing an injective b_{ji} that establishes a one-to-one mapping between elements of different domains means that we only allow domains of the same level of abstraction. While being at the same level of detail, an injective semantic relation does not restrict the coverages of the two domains. They may overlap because we neither require $b_j i$ to be non-partial nor do we assume that it covers all of the target domain. The latter is required if we want to preserve negation. In a logic where negation is defined by set difference wrt. to the complete domain it is clear that negation will only have the same effect if the domains are comparable which is only given in case of a bijective $b_j i$.

From a practical point of view, some of the assumption are more likely to hold for modular ontologies than other. While we can often assume that different modules are at the same level of abstraction, their coverage may vary as sometimes one module will just be a different view on exactly the same set of objects an sometimes they will cover completely different aspects of a domain being only related by a few concepts. Therefore, we think that defining the semantic relation to be an injective function is a good compromise between flexibility of coupling and preservation of operator semantics.

4.4 Compilation and Integrity

The bottomline of the investigations above is that in order to completely determine the subsumption relations in an ontology module that contains externally defined concepts, we might also have to perform subsumption reasoning in the modules the external concepts are mapped to. This fact contradicts the requirement of local reasoning stated in the motivation as subsumption reasoning in the external module may in turn require reasoning in modules this one is linked to an so on. In order to reach the goal of local reasoning, we therefore have to find a way to avoid the need to look beyond the the border of a module at run time.

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 [9]. 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 reason-

ing instead of the external definitions of concepts. This set of additional axioms can be computed using Algorithm 1.

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

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:

Definition 18 (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. This problem is discussed in more details in the next section.

5 Evolution Management

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 [23]. 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.

5.1 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 3 (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 3 (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 4 *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 3 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.

5.2 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[1] knowledge model in the same style as done in [3]. 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[2]. 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

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

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

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. 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.[3] The specification of effects is not complete, in the sense that it describes

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.

"worst case" scenario's, and that for some operations the effect is "unknown" (i.e. unpredictable). In contrast to [17] who provides complete semantics of changes we prefer to use heuristics in order to avoid expensive reasoning about the impact of changes.

5.3 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

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

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 [26]. 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.

6 Application in a Case Study

In order to support the claims made about the advantage of modular ontologies, we applied our model in a case study that has been carried out in the course of the WonderWeb project. Our main intend was to show that the update management procedure presented in the last section can be used to avoid the recomputation of subsumption relations in many cases. For this purpose, we defined a small example ontology using mappings to a Human Resource ontology that was developed stepwise in the case study. We used the changes that occured in the human resource 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.

6.1 The WonderWeb Case Study

The WonderWeb case study assumes that an existing database schema that is used as the basis for an ontology that should function on the Semantic Web. A database in the Human Resource (HR) domain is used as an example. The first version of the ontology is created by a tool that automatically converts a schema into an ontology [33]. In the next phase, the quality of the ontology is improved by relating this ontology to the foundational ontology DOLCE [18]. First, the HR ontology is aligned with the DOLCE ontology, and in several successive steps the resulting ontology is further refined. During this process, the ontology changes continuously, which cause problems when other ontologies refer to definitions in the evolving ontology. Therefore, in our case study, evolution management is important during the entire life-cycle of the ontology development process.

Besides this DOLCE+HR ontology, we assume that we have another ontology (we call it the *local ontology*) that uses terms and definitions from the evolving DOLCE+HR ontology (the *external ontology*). As an example, we define a very simple ontology about employees (see Figure 1). Our example ontology introduces the concept 'FulltimeEmployee' and defines a superclass 'Employee' and two subclasses 'DepartmentMember' and 'HeadOfDepartment' using terms from the DOLCE+HR ontology.

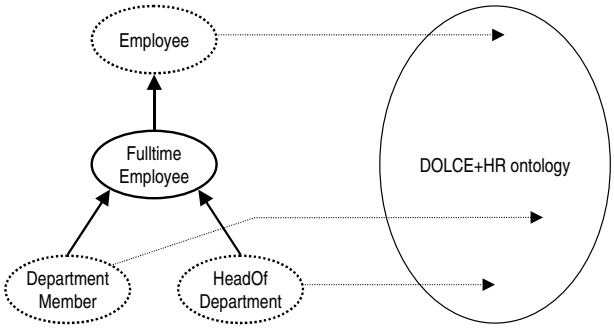

Figure 1: A simple ontology (left) with some concepts (dashed ovals) that are defined using terms from the DOLCE+HR ontology (schematically representation by al large oval).

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 the first step, the extracted HR ontology is aligned with the DOLCE foundational ontology, i.e. the concepts and properties in the HR ontology are connected

to concepts and properties in the DOLCE ontology via subsumption relations. For example, the concept ‘Departments’ from the HR ontology is made a subclass of ‘Social-Unit’ in DOLCE.

- The refinement step involves a lot of changes. Some property restrictions are added, and some additional concepts and properties are created to define the HR concepts more precisely. For example, the concept ‘Administrative-Unit’ is introduced as a new subclass of ‘Social-Unit’, and the concept ‘Departments’ is made a subclass of it. Also, the range of the property ‘email’ is restricted from ‘Abstract-Region’ to its new subclass ‘Email’.

- In the next step, a number of concepts and properties are renamed to names that better reflect their meaning. For example, ‘Departments’ is renamed to ‘Department’ (singular), and the two different variants of the relation ‘manager_id’ are renamed to ‘employee_manager’ and ‘department_manager’.

- In the final step, the tidying step, all properties and concepts that are not necessary anymore are removed and transformed into property restrictions. For example, the property ‘employee_email’ is deleted and replaced by an existential restriction in the class ‘Employee’ on the property ‘abstract_location’ to the class ‘Email’.

6.2 Modularization in the Case Study

If we now consider the problem statement from the case study, we have an local ontology with a concept hierarchy that is built up by the following explicitly stated subsumption relations (see Figure 1 again):

$$\begin{aligned} FulltimeEmployee &\sqsubseteq Employee \\ DepartmentMember &\sqsubseteq FulltimeEmployee \\ HeadOfDepartment &\sqsubseteq FulltimeEmployee \end{aligned}$$

This ontology introduces ’Full time employee’ as an 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:

$$Employee \equiv HR : Employee(x)$$

Another concept that is externally defined in the ’Head of Department’ concept. We define it to be the set of all instances that are in the range of the ’department manager’ relation.

The definition of this view given below shows that our approach is flexible enough to define concepts in terms of relations.

$$HeadOfDepartment \equiv HR : \exists y[departmentManager(y,x)]$$

An example for a more complex external concept definition is the concept 'department member' which is defined using a query that consists of three conjuncts, claiming that a department is an employee that is in the has_member relation with a Department.

$$DepartmentMember \equiv HR : \exists y[Department(y) \wedge has]_member(y,x) \wedge Employee(x)]$$

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.

$$\models DepartmentMember \sqsubseteq Employee \tag{12}$$

At a first glance, there is no relation between the definition of a Head of Department and the other two statements as it does not use any of the concept- or relation 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 the department_manger is set to 'Department' and the domain to 'Manager', the definition of HeadofDepartment is equivalent to:

$$\exists y[Department(y) \wedge department\_manager(y,x) \wedge Manager(x)]$$

As we further know that Manager is a subclass of Employee and department_manager is a sub-relation of has_member, we can derive the following subsumption relation between the externally defined concepts:

$$\models HeadOfDepartment \sqsubseteq Employee \tag{13}$$
$$\models HeadOfDepartment \sqsubseteq DepartmentMember \tag{14}$$

When the relations 12–14 are added to the local ontology, it possible to do subsumption reasoning without having to access the DOLCE+HR ontology anymore.

6.3 Updating the Models

We will now illustrate that the conclusions of the procedure are correct by studying the impact of changes mentioned in the problem statement.

Example 1: The Employee Concept The first change we observed is the removal of properties from the Employee concept. Our rules tell that this change makes the new version more general compared to its old version:

$$Employee \sqsubseteq Employee'$$

According to our procedure, this shouldn't 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.

Example 2: The department_manager Relation The second example, we have to deal with a change affecting a relation that is used in en external definition. The relation department_manager is specialized by restricting its range to a more specific concept making it a subrelation of its previous version:

$$department\_manager \sqsupseteq department\_manager'$$

Again, this is harmless according to our procedure, as department_manager is in the 'subsumed list'.

The analysis show that this change has an impact on the definition of the concept HeadOfDepartment as it restricts the allowed objects to the more specific Class Manager. The new definition $HeadOfDepartment'$ is more specific that the old one: $HeadOfDepartment' \sqsubseteq HeadOfDepartment$. As the old version was already more specific than the definition of DepartmentMember and the department_manager relation is not used in the definition of the latter the implied subsumption is indeed still valid.

Example 3: The Department Concept The different changes of the definition of the department concept left us with no clear idea of the relation between the old and the new version. In this specific case, however, we can still make assertions about the impact on implied subsumption relations. The reason is that the concept occurs in both definition. Moreover, it plays the same role, namely restricting the domain of the relation that connects an organizational unit with the set objects that make up the externally defined concept. As a consequence, the changes have the same impact on both definitions thus not invalidating the implied subsumption relation.

7 Discussion

In this deliverable, we discussed an infrastructure for representation and reasoning with modular ontologies. The intend 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 ontlogy representation for the semantic web:

1. We presented a formal model for describing dependencies between different ontologies. We proposed conjunctive queries for defining concept 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 existing standard, in particular 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 modelling 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.

In summary, this deliverable covers the representation of modular ontologies on a syntactic and semantic level as well as the notion of logical consequence as a basis for inferencing. The notions defined here can be exploited by knowledge engineers to design newly created ontologies in a modular fashion. What is still missing in order to support a wide adoption of this infrastructure are methods that support the process of migrating existing ontologies to this new infrastructure. As the way of defining concepts we propose is equivalent to OWL, the missing part is a set of methods that analyze ontologies and split them up into modules according to the principles of maximal internal cohesion and maximal external independence. A number of such methods are known from the area of object oriented databases, where so called fragmentation methods are used to determine an optimal distribution of object definitions over different object bases. Further, In the area of parallel processing algorithms for partitioning graphs into a set of subgraphs have been developed that could be applied to ontologies when regarding the RDF encoding of the ontology as a graph that has to be split up. Such methods for identifying modules are not only interesting for splitting up existing ontologies, they can also be used as a design tool to help knowledge engineers to come up with a useful set of modules.

References

[1] E. Amir and S. McIlraith. Partition-based logical reasoning. In *7th International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, 2000.

[2] 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.

[3] J. Banerjee, W. Kim, H.-J. Kim, and H. 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.

[4] S. Bechofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A reason-able ontology editor for the semantic web. In F. Baader, G. Brewka, and T. Eiter, editors, *KI 2001: Advances in Artificial Intelligence*, pages 396–408. Springer, 2001.

[5] A. Borgida and L. Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In R. Meersman and Z. Tari, editors, *On The Move to Meaningful Internet Systems 2002: CoopIS, Doa, and ODBase*, volume 2519 of *LNCS*, pages 36–53. Springer Verlag, 2002.

[6] D. Brickley, R. Guha, and A. Layman. Resource description framework (RDF) schema specification. Working draft, W3C, August 1998. http://www.w3c.org/TR/WD-rdf-schema.

[7] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *The Semantic Web - ISWC 2002*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002.

[8] M. Buchheit, F. D. W. Nutt, and A. Schaerf. Terminological systems revisited: Terminology = schema + views. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, 1994.

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

[10] D. Calvanese, G. De Giacomo, and M. 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.

[11] P. Clark, J. Thompson, K. Barker, B. Porter, V. Chaudhri, A. Rodriguez, J. Thomere, S. Mishra, Y. Gil, P. Hayes, and T. Reichherzer. Knowledge entry as the graphical assembly of components. In *Proceedings of the 1st International Conference on Knowledge Capture (K-Cap'01)*, 2001.

[12] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. Web ontology language (owl) reference version 1.0. Working draft, W3C, November 2002. http://www.w3.org/TR/owl-ref/.

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

[14] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[15] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Al-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems (JIIS)*, 27(1), 1998.

[16] W. Farmer, J. Guttman, and F. Thayer. Little theories. In D. Kapur, editor, *Proceedings of the Eleventh International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Computer Science*, pages 567–581. Springer Verlag, 1992.

[17] E. Franconi, F. Grandi, and F. 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.

[18] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, volume 2473 of *Lecture Notes in Computer Science*, page 166 ff, Sigüenza, Spain, Oct. 1–4, 2002.

[19] F. Giunchiglia and 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)*, pages 282–289. Morgan Kaufmann, 1998.

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

[21] V. Haarslev and R. Moller. Description of the RACER system and its applications. In *Proceedings of the Description Logics Workshop DL-2001*, pages 132–142, Stanford, CA, 2001.

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

[23] J. Heflin and J. 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.

[24] I. Horrocks. The FaCT system. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*, number 1397 in Lecture Notes in Artificial Intelligence, pages 307–312. Springer-Verlag, Berlin, May 1998.

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

[26] M. Klein, A. Kiryakov, D. Ognyanov, and D. Fensel. Ontology versioning and change detection on the web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, Sigüenza, Spain, Oct. 1–4, 2002.

[27] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50, 1988.

[28] A. Levy and M.-C. Rousset. Carin: A representation language combining horn rules and description logics. In *Proceedings of the 12th European Conf. on Artificial Intelligence (ECAI-96)*, pages 323–327, 1996.

[29] 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.

[30] N. Noy, R. Fergerson, and M. Musen. The knowledge model of protege-2000: Combining interoperability and flexibility, 2000.

[31] A. Rector. Modularisation of domain ontlogies implemented in description logics and related formalisms including owl. In *Proceedings of the 16th International FLAIRS Conference*. AAAI, 2003.

[32] R. Volz, A. Mdche, and D. Oberle. Towards a modularized semantic web. In *Proceedings of the ECAI'02 Workshop on Ontologies and Semantic Interoperability*, 2002.

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

[34] R. Volz, D. Oberle, and R. Studer. Views for light-weight web ontologies. In *Proceedings of the ACM Symposium on Applied Computing SAC 2003*, 2003.