Implementing Modular Ontologies with Distributed Description Logics

Heiner Stuckenschmidt

University of Mannheim
A5, 6, 68159 Mannheim, Germany
heiner@informatik.uni-mannheim.de

Abstract. In an earlier paper, we presented a logical framework for representing and reasoning with modular ontologies with a special focus on supporting localized reasoning and integrity in the face of changes. This framework while being based on a formal semantics, was not specific to a particular logic used to specify ontologies and links between modules. As a result, no system was provided that implemented the ideas presented in that paper. In this work, we close this gap by explaining, how the general framework for modular ontologies can be mapped onto distributed description logics and implemented using the DRAGO reasoning system. In particular, we refine the notion of modular ontologies to the case where local ontologies are represented in $\mathcal{SHIQ}$. We define a sound and complete inference rule for modular ontologies based distributed decsription logic and analyze the worst case complexity of reasoning in the framework. We also briefly discuss the DRAGO system and describe how our framework can be mapped on the representations supported by DRAGO.

1 Motivation

The benefits of modularization have been acknowledged in many area of computer science. With the increasing use of ontologies in real applications, the need for modular representations that support selective reuse, easier maintenance and efficient localized reasoning has been recognized by a number of researchers (e.g. [2], [1], [5]). In [14] we proposed a representational framework for modular ontologies and discussed the problem of maintaining integrity in a system of connected modules. In particular, we proposed to extend description logics with the notion of external concepts that are defined in terms of expressions over the vocabulary of other modules. We proposed to compile implied subsumption relations to facilitate local reasoning and defined the notion of integrity of compiled knowledge. While a number of abstract definitions and algorithms were given in that paper, there was no clear indication of how to implement the approach using existing technologies. In this paper, we fill this gap by explicitly linking the notion of modular ontologies to Distributed Description Logics were the language of the local models is $\mathcal{SHIQ}$. This enables us to use existing technology, in particular the DRAGO reasoning system [12] to implement the abstract definitions provided in [14].

This paper contains the following contributions that go beyond the work published in [14]:

- We formally relate the notion of a modular ontology to distributed description logics
- We formally characterize reasoning in modular ontologies in terms of the relation to DDL
- We show that the worst-case complexity of reasoning in modular ontologies is the same as for reasoning in the local ontologies
- We extend the compilation algorithm given in [14] to cover all possible inferences of subsumption relations
- We explain how the approach can be implemented in the DRAGO system.

The paper is structured as follows: We first briefly recall basic definitions of description logics and formally introduce the language $\mathcal{SHIQ}$. We then recall distributed description logics as introduced in [4]. Equipped with these definitions, we formally define modular ontologies in terms of a subset of DDL and provide rationales for the restrictions applied. We then discuss the problem of reasoning in DDL in general and in modular ontologies in particular stating a complete inference rule and providing teh complexity result. We also discuss the special reasoning task of compilation and provide a definition of integrity equivalent to the one presented in [14]. We finally describe how the formal frame work can be implemented in the DRAGO system and close with an in depth discussion of the design decisions made and the implications of compilation as a means for improving reasoning performance.

2 Description Logics

In this paper, we consider ontologies represented in the description logic $\mathcal{SHIQ}$. This choice is motivated by the fact that $\mathcal{SHIQ}$ covers a large part of the expressive power of the Web Ontology Language OWL [6], more specifically of the language OWL-DL, a decidable sublanguage of OWL that directly corresponds to the logic $\mathcal{SHOIQ}$ that extends $\mathcal{SHIQ}$ with nominals [9]. We omit this extension in order to be able to base our framework on recent results on Distributed Description Logics [4, 13] that provide us with basic mechanisms for specifying links between concepts in different ontologies in a loose way. Before defining our notion of modular ontologies, we briefly introduce the logic $\mathcal{SHIQ}$ as well as the basic notions of Distributed Description Logics. For further information about notation and naming in Description Logics, we refer to [3].

Let $\mathcal{C}$ be a set of concept names and RN a set of role names. Further let there be a set $R^+ \subseteq RN$ of transitive roles (i.e. for each $r \in R^+$ we have $r(x,y) \wedge r(y,z) \Rightarrow r(x,z)$). If now r^- denotes the inverse of a role (i.e. $r(x,y) \Rightarrow$

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

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

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

– $\top$ and $\bot$ are concept expressions for the most general concept and the unsatisfiable concept, respectively;
– every concept name A is a concept expression;
– if C and D are concept expressions, r is a role, s is a simple role and n is a non-negative integer, then $\neg C$, $C \sqcap D$, $C \sqcup D$,$\forall r.C$, $\exists r.C$, $(\geq n\, r.C)$ and $(\leq n\, r.C)$ are concept expressions.

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

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

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

- $(\leq n\, r.C)^{\mathcal{I}} = \{x | \#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$

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

3 Distributed Description Logics

Distributed Description Logics as proposed in [4] provide a language for representing sets of terminologies and semantic relations between them. For this purpose DDLs provide mechanisms for referring to terminologies and for defining rules that connect concepts in different terminologies. On the semantic level, DDLs extend the notion of interpretation introduced above to fit the distributed nature of the model and to reason about concept subsumption across terminologies.

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

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

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

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

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

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

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

4 A Framework for Modular Ontologies

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

4.1 Architecture

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

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

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

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

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

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

4.2 Restricting Mapping Expressiveness

In Distributed Description Logics, there are no restrictions on the the antecedent of a bridge rule—except that it has to be a valid concept of the source terminology. In our framework, we restrict this freedom for the sake of an easier maintenance of the semantic relations between terminologies. This restriction is motivated by our earlier work on keeping integrity in modular ontologies reported in [14]. In that work, we proposed a heuristic approach for determining the impact of changes in other modules on the correctness of local subsumption reasoning. The approach relied on the fact that changes were only monotonically propagated to other modules. In order to achieve this effect also in the framework of distributed description logics, we restrict the language used to specify externally defined concepts to a sublanguage of $\mathcal{SHIQ}$ that does not contain operators that can have a non-nonotonic effect, in particular negation, universal restrictions and qualified number restrictions that limit the number of related concepts. More precisely, we allow concept expressions that are defined in the following sublanguage of $\mathcal{SHIQ}$:

$$C, D \longrightarrow \top \,|\, \bot \,|\, A \,|\, C \sqcap D \,|\, C \sqcup D \,|\, \exists r.C \,|\, \exists r^{-}.C \,|\, \geq n\, r.C \,|\, \geq n\, r^{-}.C$$

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

5 Reasoning in Modular Ontologies

The direct correspondence of our framework to Distributed Description Logic allows us to base inference in modular ontologies on known results for the corresponding DDL. In particular, we can provide completeness and complexity results for reasoning in modular ontologies. We extend the existing work on reasoning in DDL with the notion of compilation of implied subsumption relations. Specifically, we use reasoning methods for Distributed Description Logics to derive subsumption relations between externally defined concepts in modules and explicitly add the derived subsumption relations as axioms to the module. The results of [13] guarantee that after this compilation step reasoning can be performed locally without considering other modules unless there are changes in the system.

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

5.1 Reasoning based on DDL

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

$$\frac{i{:}A \xrightarrow{\sqsupseteq} j{:}G,\, i{:}B \xrightarrow{\sqsubseteq} j{:}H,\, i{:}A \sqsubseteq B}{j{:}G \sqsubseteq H} \tag{1}$$

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

$$\frac{i{:}A \xrightarrow{\sqsupseteq} j{:}G,\, i{:}B_k \xrightarrow{\sqsubseteq} j{:}H_k (1 \leq k \leq n),\, i{:}A \sqsubseteq \bigsqcup_{k=1}^{n} B}{j{:}G \sqsubseteq \bigsqcup_{k=1}^{n} H_k} \tag{2}$$

It has been shown that this general propagation rule completely describes reasoning in DDLs that goes beyond well known methods for reasoning in Description Logics. To be more specific, adding the inference rule in equation 2 to existing tableaux reasoning methods leads to a correct and complete method for reasoning in DDLs. A corresponding result using a fixpoint operator is given in [13]. Based on these results, we can define a general inference rule for the case of modular ontologies in the following way:

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

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

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

Complexity As we reduce reasoning in modular ontologies to reasoning in DDLs with $\mathcal{SHIQ}$ as a local language, complexity results can be derived from known results on reasoning in $\mathcal{SHIQ}$ and Distributed Description Logics. In particuar, reasoning in modular ontologies is at least as hard as reasoning in $\mathcal{SHIQ}$: in the extreme case a modular ontology consists only of a single module without external concepts, thus reasoning in modular ontologies is equivalent to reasoning in $\mathcal{SHIQ}$. Further, reasoning in modular ontologies is not harder than reasoning in $\mathcal{SHIQ}$ because we reduce reasoning in modular ontologies to reasoning in DDLs. There exists a reduction of reasoning in DDLs with $\mathcal{SHIQ}$ as a local language to $\mathcal{SHIQ}$[**?**]. Both reductions are linear in the size of the resulting terminology and therefore do not change the complexity class. This shows that the complexity of reasoning in modular ontologies is not worse than reasoning in the web ontology language. Using the reduction of DDL to $\mathcal{SHIQ}$ it is even possible to use existing OWL reasoners for reasoning with modular ontologies. Although practical implementations of OWL reasoners have shown that good average case performance can be achieved, the worst case complexity is still very high and asks for further optimization.

5.2 Compilation and Integrity

Existing reasoners for expressive Description Logics are highly optimized with respect to deciding subsumption in the context of a single T-Box. Serafini and Tamilin present a distributed reasoning system that extends existing reasoners to distributed T-Boxes [12]. In theory, this system is complete with respect to the propagation rules described above and has—as we have argued—the same worst-case complexity. In practice, however, reasoning with multiple, possible distributed modules, brings some new problems with respect to completeness and reasoning performance. First of all the completeness of the distributed reasoners depends on the availability of local reasoners for all T-Boxes in the system. In a loosely coupled network without central control this cannot always be guaranteed as network nodes can be unreachable or even leave the network. In this case, necessary subsumption tests cannot be performed at these nodes leading to a possible incompleteness. Another problem currently not addressed in the work of Serafini and Tamilin are performance problems due to communication costs between the different nodes in the system. Work in the area of distributed databases has shown that communication costs often become serious bottlenecks in distributed systems.

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

Of course these runtime benefits have their price in terms of computational complexity of the compilation step. The completeness of the propagation rule given in equation 2 tells us that to be independent from other modules we only have to consider subsumption relations between externally defined concepts, as only such subsumption relations can be propagated from outside. What we have to check is subsumption between each external concept and the disjunction of all combinations of other external concepts. For a local module, this process is defined in algorithm 1.

Algorithm 1 compile

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

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

While the results of Serafini and others [12] guarantee that local reasoning is correct and complete at the time the compilation is carried out, a problem occurs when changes are made to the system. Changes in the definitions of the external concepts, but also changes in the definitions of concepts and relations in other modules can make local reasoning incomplete or inconsistent. In order to prevent situations in which local reasoning is not correct and complete any more we introduce the notion of integrity of a modular ontology.

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

$$\mathcal{I}_i' \models C \sqsubseteq D \Leftrightarrow \mathfrak{I} \models i : C \sqsubseteq D$$

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

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

6 Implementing Modular Ontologies in DRAGO

DRAGO is a reasoning system for Distributed Ontologies that supports reasoning in distributed description logics with the limitation that bridge rules might only be specified between concept names. The DRAGO System consists of a set of reasoners each implemented as an independent peer. Each of the peers has a local ontology represented in the subset of OWL-DL that corresponds to the description logic $\mathcal{SHIQ}$ (i.e. OWL-DL without enumerated classes and datatypes). The different peers are connected to form a peer network by means of mappings (sets of bridge rules) connecting the individual ontologies and forming a distributed T-Box [12]. Given a distributed T-Box, $\mathfrak{T}$, the following reasoning services are available in the DRAGO system:

- *Local/global satisfiability:* check if $\mathcal{T}_i \models C \equiv \bot$, and $\mathfrak{T} \models i\!:\!C \equiv \bot$
- *Local/global subsumption:* check if $\mathcal{T}_i \models C \sqsubseteq D$, and $\mathfrak{T} \models i\!:\!C \sqsubseteq D$
- *Local/global classification:* Produce a classification on the atomic concepts of T_i. A classification on a set of atomic concepts $\mathcal{C}$, is directed acyclic graph $\langle \mathcal{C}, \prec, \sim \rangle$, where $\mathcal{C}$ is the set of atomic concepts of the language of $\mathcal{T}_i$ and $\prec$ constitute a directed acyclic graph on $\mathcal{C}$, and $\sim$ is an equivalence relation on $\mathcal{C}$. And the following properties holds, $C \sim D$ iff $\mathcal{T}_i \models C \equiv D$ (resp

$\mathfrak{T} \models i{:}C \equiv D$), $C \prec D$ if and only if $\mathcal{T}_i \models C \sqsubseteq D$ and $\mathcal{T}_i \not\models C \sqsubseteq D$, (resp. $\mathfrak{T} \models i{:}C \sqsubseteq D$ and $\mathfrak{T} \not\models i{:}C \sqsubseteq D$). Furthermore if $C \prec D$ then for no $E \in \mathcal{C}$, $C \prec E \prec D$

Based on the correspondence between our notion of modular ontologies and a subset of distributed description logics, we can use the functionality of the DRAGO system to implement reasoning in modular ontologies. In the following, we briefly explain the steps necessary for implementing a modular ontology using the DRAGO system.

step 1: Distributing Ontology Modules In the first step, we instantiate a reasoning peer for each module of the ontology. For each peer an ontology is created consisting of the internal concept definitions of the respective module.

step 2: Create Mappings For each pair of modules M_i, M_j we create the mapping set. This is done by looking at the external concept definitions in module M_i. For each external defintion $i : C \equiv M_j : D$ we create a new concept name D (note that in the external defintion D' can be a complex concept expression) and add the two bridge rule $i : C \xrightarrow{\sqsubseteq} j : D'$ and $i : C \xrightarrow{\sqsupseteq} j : D'$ to the mapping set $\mathfrak{B}_{ij}$.

step 3: Enrich Local Ontologies For externally defined concept $i : C \equiv M_j : D$, we create a new concept definition $D' \equiv D$ and add it to the ontology created for module M_j. This step is necessary for cope with the limitation of DRAGO system to mappings between concept names.

step 4: Connect Local Ontologies Finally, we connect the different modules by loading the mappings created in step 2 into the system thereby connecting the different modules.

This implementation of modular ontologies allows us to use the reasoning services provided by DRAGO to compute implied subsumption relations and to store them locally to support local reasoning as explained in the previous sections. If we restrict our approach to the simplified inference relation shown in equation 1 we can directly use the global classification method of DRAGO to compute all implied subsumption relations and store them locally. If the goal is completeness, we have to add another control structure that checks subsumption between a concept and all combinations of disjunctions of other concept names. This is computationally quite expensivem but can easily be done using the DRGAO API.

7 Conclusions

Reasoning in modular ontologies is complex. We have seen that the complexity is essentially the same as for reasoning in classical Description Logics which are the basis for OWL. We cannot escape this complexity, but we can move parts of the reasoning effort offline by compiling implied subsumption relations as described in Section 5.2. This approach, however, is only feasible if there are phases where the offline computation necessary to compile the implied relations can be done without affecting the performance of the system. Typically, such computations are done ‘overnight’ when the system load can be assumed to be low. An alternative for situations where this approach is not possible is to do the compilation on the fly. In particular, we can compile implied subsumption relations whenever they are computed in order to answer a user query to the system. This kind of ‘lazy compilation’ has the advantage that the enormous effort for compiling implied knowledge is done as part of the normal reasoning process. In the beginning, users will not benefit much from this approach, but the time savings increase with each query answered. In this way, we also prevent the compilation of knowledge that is never used.

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

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

A choice that is less obvious is Distributed Description Logics as a basis for the semantics of mappings. In a recent survey, we compared different approaches for describing mapping semantics [11]. One result of this comparison was that Distributed Description Logics provide the highest degree of de-coupling between different T-Boxes. This is important for our purposes as we want to support localized reasoning. A generalization of Distributed Description Logics in terms of a distributed version of first order logic has been described by Serafini and Ghididi [8]. We could have chosen this more general framework as the basis for our work, however, the drawback of this is the lack of existing reasoning methods. Distributed Description Logics come with a well investigated and implemented proof system that can be used to implement our approach.

The most controversial choice is to restrict the language that can be used to define external concepts. The framework of Distributed Description Logic allows us to use arbitrary $\mathcal{SHIQ}$ expressions in the definitions. A corresponding more general approach would have the same properties with respect to logical consequence, compilation and local reasoning. The restriction to the sublanguage was motivated by the importance of the monotonicity property for the definition of update heuristics defined in [14]. This means that external concepts can be defined using a more expressive language. This, however, would come at the price that implied subsumption relations concerning this concept would have to be recomputed every time a change occurs. We believe that the restriction proposed in this paper is reasonable as it allows the use of update heuristics.

References

1. Philippe Adjiman, Philippe Chatalic, Francois Goasdoue, Marie-Christine Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic we. *Journal of Artificial Intelligence Research*, 25:269–314, 2006.
2. E. Amir and S. McIlraith. Partition-based logical reasoning. In *7th International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, 2000.
3. 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.
4. A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003.
5. Bernardo Cuenca-Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*, 2006.
6. M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein. Web ontology language (owl) reference version 1.0. Working draft, W3C, November 2002. http://www.w3.org/TR/owl-ref/.
7. Richard Fikes, Patrick Hayes, and Ian Horrocks. Owl-ql: A language for deductive query answering on the semantic web. *Journal of Web Semantics*, 2(1), 2005.
8. Chiara Ghidini and Luciano Serafini. Distributed first order logic - revised semantics. Technical report, ITC-irst, January 2005.
9. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, Lecture Notes in Computer Science. Springer Verlag, 2000.
10. Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.

11. L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping languages for terminological knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence - IJCAI05*, Edingurgh, UK, August 2005.
12. L. Serafini and A. Tamilin. DRAGO: Distributed reasoning architecture for the semantic web. In *In Proceedings of the Second European Semantic Web Conference (ESWC'05)*. Springer-Verlag, 2005.
13. Luciano Serafini, Alex Borgida, and Andrei Tamilin. Aspects of distributed and modular ontology reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI-05*, Edinburgh, Scotland, 2005.
14. H. Stuckenschmidt and M. Klein. Integrity and change in modular ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'03*, pages 900–905, Acapulco, Mexico, 2003. Morgan Kaufmann.
15. Stephan Tobies. *Complexity results and practical algorithms for logics in Knowledge Representation.* Phd thesis, LuFG Theoretical Computer Science, RWTH-Aachen, 2001.
16. Jeffrey D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.