Reasoning about Mappings in Distributed Description Logics

Bachelor Thesis

presented by
Christian Meilicke
Bensheim

submitted to the
Knowledge Representation and Knowledge Management Research Group
Prof. Dr. Heiner Stuckenschmidt
University Mannheim

August 2006

Supervisor:
Prof. Dr. Heiner Stuckenschmidt

Contents

List of Algorithms

List of Figures

List of Tables

Part I

1 Motivation

An essential element of the semantic web is the use of ontologies to describe the domain of an information source. As argued by Baader, Horrock, and Sattler in [2], description logics are well-suited as ontology languages. Describing an information source by defining an ontology of the domain, results in the problem of semantic integration. If the same domain is described by different terminologies, there are also different corresponding domain representations that have to be integrated (compare [4]). In distributed description logics semantic mappings are used to bridge the gap between different ontologies.

Stuckenschmidt, Serafini, and Wache distinguish two lines of work related to the problem of semantic integration (see [9]). On the one hand research focuses on the use of semantic mappings for reasoning and query answering. On the other hand tools for the (semi-)automatic generation of semantic relations are developed. Obvioulsy, a framework to describe semantic mappings from a formal point of view will be useful for distributed reasoning as well as for automatic mapping generation. Therefore, a number of formal mapping properties are presented by the authors of [9].

In this thesis the approach of Stuckenschmidt, Serafini, and Wache will be resumed and extended. Especially the following questions will be raised.

- **Theoretical Framework** Are there further interesting mapping properties? Can useful dependencies between mapping properties be detected?
- **Algorithms** Are there efficient algorithms for deciding mapping properties? How can these algorithms be implemented?
- **Mapping Diagnosis** How can mapping properties be applied in the context of mapping diagnosis and mapping repairing?
- **Runtime** It is possible to optimize the runtime of distributed reasoning by using mappings that have been modified according to some of the properties?

These questions will be answered in part II and part III of this thesis. But first of all in section 2 a short introduction into distributed description logics is given to explain how semantic mappings are realized as sets of bridge rules. Answering the last two questions is not possible by pure theoretical investigations. Therefore, a tool for computing mapping properties has been developed as part of this thesis.[1]

[1] See appendix A for further information.

This tool makes use of the DRAGO system that is introduced in section 3. The first part ends in section 4 with an outline of the following contents. In this section the author explains his contribution with respect to the research questions listed above.

2 Distributed Description Logics

In the following it is assumed that the reader is familiar with description logics. An introduction can be found in [3]. Distributed description logics, as described by Serafini and Tamilin in [7], can be understood as a framework for formalization of multiple terminologies pairwise linked by directed semantic mappings. In this context a pair of terminologies $\mathbb{T}$ and associated mappings $\mathbb{M}$ is called a distributed terminology $\mathfrak{T} = \langle \mathbb{T}, \mathbb{M} \rangle$.

Let I be set of indices. Then $\mathbb{T} = \{\mathcal{T}_i\}_{i \in I}$ denotes the set of all terminologies in $\mathfrak{T}$ and $\mathcal{T}_i$ with $i \in I$ denotes the i-th terminology of $\mathfrak{T}$. Each terminology $\mathcal{T}_i$ is a T-Box of a description logic theory. Therefore, it contains definitions of concepts and properties as well as axioms relating concepts and properties to each other. To refer without ambiguity to a concept C from terminology $\mathcal{T}_i$, the index of the terminology is used in front of the concept, for example $i : C$.

$\mathbb{M} = \{\mathcal{M}_{ij}\}_{i \neq j \in I}$ refers to the mappings of $\mathfrak{T}$. A mapping $\mathcal{M}_{ij}$ is a set of bridge rules that establishes semantic relations from $\mathcal{T}_i$ to $\mathcal{T}_j$. Every bridge rule in $\mathcal{M}_{ij}$ has a certain type and connects a concept from $\mathcal{T}_i$ to a concept from $\mathcal{T}_j$. The following three types of bridge rules are known to the DRAGO system that will be introduced in the next section.

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

Bridge rules from $\mathcal{T}_i$ to $\mathcal{T}_j$ allow a partial translation of $\mathcal{T}_i$'s language into the language of $\mathcal{T}_j$. For example, the into bridge rule $i : C \xrightarrow{\sqsubseteq} j : D$ states that concept $i : C$ is, from $\mathcal{T}_j$'s point of view, less general than or as general as concept $j : D$. The analogous onto bridge rule states that $i : C$ is more general than or as general as $j : D$. An equivalence bridge rule is the conjunction of into and onto bridge rule.

The first element of the semantics of distributed description logics is a local interpretation $\mathcal{I}_i$ for each terminology $\mathcal{T}_i$. Each interpretation $\mathcal{I}_i$ consists of a local domain $\Delta^{\mathcal{I}_i}$ and a valuation function $\cdot^{\mathcal{I}_i}$. The valuation function mapps concepts on subsets of $\Delta^{\mathcal{I}_i}$ and properties on subsets of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$. The second element is a domain relation r_{ij} that connects for each pair of terminologies $\langle \mathcal{T}_i, \mathcal{T}_j \rangle_{i \neq j}$ elements of the interpretation domains $\Delta^{\mathcal{I}_i}$ and $\Delta^{\mathcal{I}_j}$. $r_{ij}(x)$ is used to denote

$\{y \in \Delta^{\mathcal{I}_j} | (x,y) \in r_{ij}\}$ and $r(D)$ is used to denote $\bigcup_{x \in D} r_{ij}(x)$ for any $x \in \Delta^{\mathcal{I}_i}$ and any $D \subseteq \Delta^{\mathcal{I}_i}$. The pair of both elements $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I} \rangle$ is called the distributed interpretation. A distributed interpretation $\mathfrak{I}$ satisfies a distributed terminology $\mathfrak{T}$ iff for all $i \neq j \in I$ the following clauses are true.

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

Due to the introduction of bridge rules it is possible to transfer knowledge between different terminologies that changes subsumption relations in the target terminology. Consider terminology $\mathcal{T}_1$ with axiom (1).

$$1 : A \sqsubseteq B \tag{1}$$

and a second terminology $\mathcal{T}_2$ with two concepts C and D. Let $\mathcal{M}_{ij}$ consist of bridge rules (2) and (3).

$$1 : A \xrightarrow{\sqsupseteq} 2 : C \tag{2}$$

$$1 : B \xrightarrow{\sqsubseteq} 2 : D \tag{3}$$

Since every model for (1), (2), and (3) is also a model for (4), (4) follows by distributed reasoning.

$$2 : C \sqsubseteq D \tag{4}$$

This short introduction into distributed description logics should be sufficient for the understanding of the following sections. A more detailed description can be found in [5]. In the next section it will be explained how distributed reasoning is implemented in the DRAGO system.

3 DRAGO

DRAGO is a peer-to-peer system for reasoning with multiple terminologies interconnected by semantic mappings.[2] The acronym 'DRAGO' is an abbreviation for 'Distributed Reasoning in a Galaxy of Ontologies'. A more detailed system description can be found in [7].

[2]I thank Andrei Tamilin for his assistance with DRAGO and many interesting discussions on related topics.

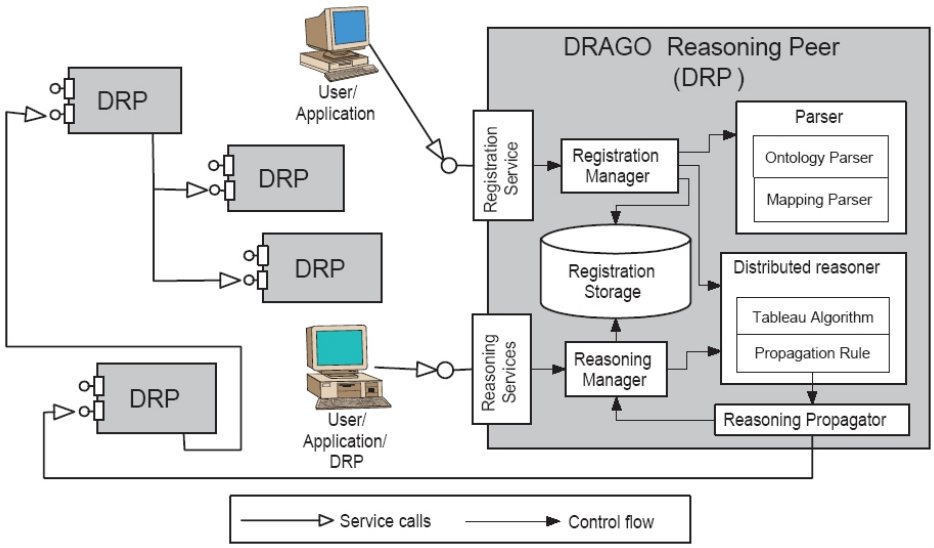

Figure 1: Architecture of a DRAGO reasoning peer (taken from [7]).

The major component of the DRAGO system is the DRAGO Reasoning Peer (DRP). If a distributed terminology $\mathfrak{T}$ is instantiated as DRAGO network, for each terminology $\mathcal{T}_i$ of $\mathfrak{T}$ there exists a DRP offering reasoning services for $\mathcal{T}_i$. The components and interfaces of a DRP are depicted in figure 1. A DRP offers two interfaces. Before starting a DRP the registration service interface is used to register a terminology $\mathcal{T}_j$ and associated mappings. For each mapping $\mathcal{M}_{ij}$ the address of the DRP hosting terminology $\mathcal{T}_i$ has to be specified. Note that mapping $\mathcal{M}_{ij}$ from $\mathcal{T}_i$ to $\mathcal{T}_j$ is registered at the peer hosting terminology $\mathcal{T}_j$. Having started a DRP, the reasoning service interface enables calling reasoning services concerned with the hosted terminology. Using these services it is possible to execute the following operations:

- Check consistency of a terminology.
- Compute the taxonomy of a terminology.
- Check entailment.
- Check satisfiability of a certain concept.

In section 6.3 it will be argued that there are two groups of operations in so far as we are interested in the runtime of the operations.

Every reasoning service can be used in one of two modes. It is, for example, possible to build a local taxonomy as well as a distributed taxonomy. For any local

operation a standard tableau reasoner is used. In this case the results of reasoning are in no way affected by the available mappings. In opposite to local operations in distributed mode reasoning requests are propagated to other DRPs according to the distributed tableau algorithm (see [8] and [7] for more information on the distributed tableau algorithm).

The DRAGO system provides the basic functionality for implementing the algorithms presented below. For some algorithms it is sufficient to use the functionality of the reasoning service interface. For the more complex algorithms temporary changes have to be applied to mappings and ontologies. Thus, registration services have to be used in accordance with the results of the reasoning requests. A detailed description can be found in section 6.2.

4 Outline and Contributions

In part II a theoretical framework of mapping properties will be developed and corresponding algorithms for deciding these properties will be stated. For that reason the basics of distributed description logic have been explained in section 2. Part III will continue with an experimental evaluation and will shows how the defined properties can be used to describe mappings. The experiments have been realized with a tool that is based on the services of the DRAGO system. Therefore, the DRAGO system has been introduced in section 3. Since the groundwork for part II and III has been done, we can now take a closer look at the contents of the following sections.

Section 5 introduces the mapping properties that constitute the main subject of this thesis. The properties of consistency, embedding, entailment, and minimality have already been described in [9]. In addition several new properties are introduced. These are the properties of stability, invasiveness, and irreducibility. Besides defining additional mapping properties, one objective of the thesis is to find dependencies between mapping properties. Several propositions are stated to describe these dependencies. The section ends with a classification of the introduced properties according to two dimensions.

In section 6 for each property an algorithm is stated that checks wether or not the property holds. Four algorithms have already been stated partially in [9]. Refining these algorithms and adding additional algorithms is another contribution of this thesis. All algorithms have been implemented by the author using the services of the DRAGO system. A short description of this implementation is given. Finally, the worst case runtimes of the algorithms are compared with respect to their usage of DRAGO reasoning services.

The experimental evaluation is described in section 7. Thus, attention is fo-

cused no longer on theoretical questions but on questions of applicability. After describing the ontologies and automatically generated mappings chosen as test-cases, the defined properties are used for mapping diagnosis. By applying the implemented algorithms formal defects can be detected in many generated mappings. This shows that the theoretical framework provides a powerful tool in mapping diagnosis. This approach is carried on towards mapping repairing and explained with an example.

There is some evidence that minimized and reduced mappings can be used to optimize the runtime performance of standard reasoning tasks. This is verified by the experimental results of a second test series. Thus, the defined mapping properties can not only be applied in the context of mapping diagnosis and debugging, but also for optimizing the runtime of distributed reasoning.

The results of this thesis are summarized and discussed in section 8 with respect to the questions stated in the first section. Besides presenting results, three points of criticism are dealt with leading to further interesting areas of research.

Part II

5 Defining Mapping Properties

First of all a few terminological simplifications have to be introduced. Whenever an indexed expression like $\mathcal{T}_i$ is used, it is supposed that $\mathcal{T}_i$ is an element of $\mathbb{T}$ without explicitly mentioning the additional constraint $i \in I$. The same holds for an expression like $\mathcal{M}_{ij}$ and the set of mappings $\mathbb{M}$. Using an expression like $i : C$, it is also supposed that there exists a terminology $\mathcal{T}_i$ in $\mathbb{T}$. These agreements simplify the definitions and explanations presented in the following sections.

5.1 Preliminary Definitions

In section 2 a distributed terminology $\mathfrak{T} = \langle \mathbb{T}, \mathbb{M} \rangle$ has been defined as a set of terminologies $\mathbb{T} = \{\mathcal{T}_i\}_{i \in I}$ and a set of mappings $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i \neq j \in I}$. This implies by definition that every terminology and every bridge rule of $\mathfrak{T}$ is known. Concerning an implementation of a distributed terminology as a DRAGO peer network, this kind of omniscient point of view must be refrained. Each peer offering reasoning services for a terminology has direct access only to a (small) subset of the whole network, in particular to the peers in the neighbourhood. Therefore, it makes sense first to examine the structure of a distributed terminology, before introducing mapping properties in the following sections.

Since requesting reasoning services is always bound to a certain terminology, the resulting information might also - in a yet unspecified way - be only information about a subset of the distributed terminology $\mathfrak{T}$ and its mappings. To explain this in detail, it is useful to describe a distributed terminology $\mathfrak{T}$ as a directed graph (compare [11]). A directed graph $G = \langle V, E \rangle$ is defined as a pair $\langle V, E \rangle$ with V being a finite set and E being a binary relation on V. The vertex set V obviously coincides with the set $\{\mathcal{T}_i\}_{i \in I}$. The edge set E depends on the set $\{\mathcal{M}_{ij}\}_{i \neq j \in I}$ in the following way: For all pairs $\langle i, j \rangle_{i \neq j \in I}$ the edge $\langle \mathcal{T}_i, \mathcal{T}_j \rangle$ is an element of E if and only if $\mathcal{M}_{ij}$ is a non-empty set. The direction of the edge $\langle \mathcal{T}_i, \mathcal{T}_j \rangle$ is equal to the direction of the bridge rules in $\mathcal{M}_{ij}$ since the bridge rules in $\mathcal{M}_{ij}$ are mapping concepts from $\mathcal{T}_i$ into respectively onto concepts from $\mathcal{T}_j$. Thus, knowledge is transferred from $\mathcal{T}_i$ to $\mathcal{T}_j$ and $\mathcal{T}_i$ can be seen as knowledge source for $\mathcal{T}_j$. The following definition captures this conception and distinguishes between direct and indirect access to a knowledge source.

Definition 1 (Known source and source) *Given $\mathfrak{T}$, a terminology $\mathcal{T}_i$ is a known source of terminology $\mathcal{T}_j$ iff $\mathcal{M}_{ij}$ is a non-empty set. A terminology $\mathcal{T}_i$ is a source of $\mathcal{T}_j$ iff $\mathcal{T}_i$ is a known source of $\mathcal{T}_j$ or if there exists a $k \in I$ with $k \neq i$ and $k \neq j$ such that $\mathcal{T}_i$ is a source of $\mathcal{T}_k$ and $\mathcal{T}_k$ is a source of $\mathcal{T}_j$.*

Alternatively, terminology $\mathcal{T}_i$ can be defined as a known source of terminology $\mathcal{T}_j$ if and only if vertex $\mathcal{T}_j$ is adjacent to vertex $\mathcal{T}_i$. Similarly, it can be said that terminology $\mathcal{T}_i$ is a source of terminology $\mathcal{T}_j$ if and only if there exists a path from vertex $\mathcal{T}_i$ to $\mathcal{T}_j$. Figure 2 depicts an example of a distributed terminology. In this example $\mathcal{T}_4$ and $\mathcal{T}_5$ are known sources of $\mathcal{T}_3$ while both of them are sources but not known sources of $\mathcal{T}_1$.

The source-relation can be used to introduce the property of being a root terminology. $\mathcal{T}_j$ is a root terminology if $\mathcal{T}_j$ is reachable from every other terminology $\mathcal{T}_i$. If the mappings of $\mathfrak{T}$ contain cycles, a distributed terminology $\mathfrak{T}$ can have several root terminologies. On the other hand, not even one of $\mathfrak{T}$'s terminologies might be a root terminology. In figure 2 there is exactly one root terminology $\mathcal{T}_1$.

Definition 2 (Root terminology) *Given $\mathfrak{T}$, $\mathcal{T}_i$ is a root terminology iff for every $j \in I$ with $j \neq i$ terminology $\mathcal{T}_j$ is a source of $\mathcal{T}_i$.*

A request sent to a peer hosting terminology $\mathcal{T}_i$ will never be forwarded to a peer hosting terminology $\mathcal{T}_j$ if $\mathcal{T}_j$ is not a source of $\mathcal{T}_i$. Thus, the notion of a source allows us to eliminate irrelevant terminologies and mappings by reducing the whole distributed terminology to a restricted terminology.

Definition 3 (Restricted terminology) *Given $\mathfrak{T}$ and terminology $\mathcal{T}_i$, let $J \subseteq I$ be a set of indices such that for every $j \in J$ terminology $\mathcal{T}_j$ is a source of $\mathcal{T}_i$. Then $\langle \{\mathcal{T}_i\}_{i \in J}, \{\mathcal{M}_{ij}\}_{i \neq j \in J} \rangle$ is called $\mathfrak{T}$ restricted by $\mathcal{T}_i$.*

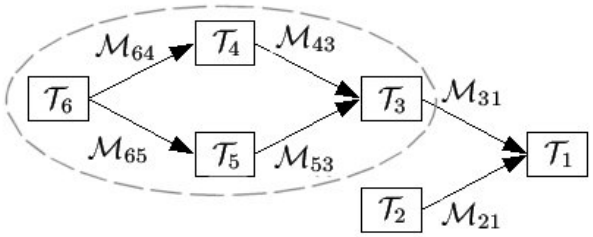

Figure 2: Example of a bridgegraph.

In figure 2 the distributed terminology restricted by $\mathcal{T}_3$ is marked by a dashed line. Whenever a certain mapping property is checked, a connection to a peer hosting one of $\mathfrak{T}$'s ontologies, say for example $\mathcal{T}_3$, has to be established. The mappings outside the distributed terminology restricted by $\mathcal{T}_3$ are unknown to the peer hosting $\mathcal{T}_3$ and have therefore no effect on the results of reasoning. The mappings within the restricted terminology can be subdivided into two groups. $\mathcal{M}_{43}$ and $\mathcal{M}_{53}$ establish a connections between $\mathcal{T}_3$ and one of its known sources in opposite to $\mathcal{M}_{64}$ and $\mathcal{M}_{65}$. Mappings of the first group are affecting the distributed taxonomy of $\mathcal{T}_3$ directly. Mappings of the second group have an influence on the distributed taxonomy of $\mathcal{T}_3$'s sources, and thus, via the mappings of the first group, an indirect influence on the distributed taxonomy of $\mathcal{T}_3$. But these mappings are not even known to the peer offering reasoning services for $\mathcal{T}_3$.

In the following subsection mapping properties like 'consistency with respect to a terminology' will be defined. If such a definition is used to express, for example, that the mappings are consistent with respect to $\mathcal{T}_3$, the truth of the statement depends fully on the distributed terminology restricted by $\mathcal{T}_3$. Anyhow, consistency with respect to $\mathcal{T}_3$ does not entail consistency with respect to any other terminology in the distributed terminology restricted by $\mathcal{T}_3$. The same holds for any other mapping property.

Checking wether a certain property holds with respect to all terminologies in $\mathfrak{T}$, requires an iterative or recursive approach in which all of $\mathfrak{T}$'s terminologies are traversed. For every visited terminology $\mathcal{T}_i$ it has to be checked if the property holds with respect to $\mathcal{T}_i$. In order to enable traversing every terminology a connection to a root terminology has to be established. Otherwise it is only possible to make statements about subsets of the whole distributed terminology.

5.2 Consistency and Stability

Suppose that there is a library using two terminologies $\mathcal{T}_1$ and $\mathcal{T}_2$. $\mathcal{T}_1$ contains information about furniture and capacity, while $\mathcal{T}_2$ is about available literature. Since the mappings between these ontologies have been developed by a bad paid student assistant, there are a few defective bridge rules. Among others we have bridge rules (5) and (6).

$$1 : bookshelf \xrightarrow{\sqsupseteq} 2 : book \tag{5}$$

$$1 : shelf \xrightarrow{\sqsubseteq} 2 : unreadable \tag{6}$$

Further, let $\mathcal{T}_1$ describe bookshelves as a subclass of shelves. By applying (5) and (6) we can conclude that in the distributed taxonomy of $\mathcal{T}_2$ books are unreadable things. But suppose that books and unreadable things have been defined as disjoint, according to our intuition. Then concept $book$ becomes distributed unsatisfiable and therefore $\mathcal{M}_{12}$ is, according to the following definition, called an inconsistent mapping. In general the mappings of a distributed terminology can be defined as inconsistent with respect to a concept $i : C$ if the additional constraints induced by the mappings have the (unintended) effect of making the local satisfiable concept $i : C$ distributed unsatisfiable. If such an effect does not occur the mappings are consistent with respect to $i : C$.

Definition 4 (Consistency) *Given $\mathfrak{T}$, $\mathbb{M}$ is consistent with respect to $i : C$ iff $\mathcal{T}_i \not\models C \sqsubseteq \bot \Rightarrow \mathfrak{T} \not\models i : C \sqsubseteq \bot$. Otherwise $\mathbb{M}$ is inconsistent with respect to $i : C$. $\mathbb{M}$ is consistent with respect to $\mathcal{T}_i$ iff for all $i : C$ $\mathbb{M}$ is consistent with respect to $i : C$. Otherwise $\mathbb{M}$ is inconsistent with respect to $\mathcal{T}_i$.*

Obviously, inconsistency is a clear symptom for defective parts in the mappings of a distributed terminology. Nevertheless, it was typical to the results of the experimental evaluation that many of the automatically generated mappings are inconsistent (compare section 7.1).

Inconsistency can be seen as a special case of the more general notion of instability. An inconsistent mapping turns a local satisfiable concept C into a distributed unsatisfiable concept, while an instable mapping imposes additional constraints on C which do not necessarily lead to a distributed unsatisfiable concept. The notion of stability can thus be defined as follows.

Definition 5 (Stability) *Given $\mathfrak{T}$, $\mathbb{M}$ is stable with respect to $i : C$ iff for all $i : D$ we have $\mathfrak{T} \models i : C \sqsubseteq i : D \Rightarrow \mathcal{T}_i \models C \sqsubseteq D$. Otherwise $\mathbb{M}$ is instable with respect to $i : C$. $\mathbb{M}$ is stable with respect to $\mathcal{T}_i$ iff for all $i : C$ $\mathbb{M}$ is stable with respect to $i : C$. Otherwise $\mathbb{M}$ is instable with respect to $\mathcal{T}_i$.*

The negative connotation of the term instability should not imply that the property of instability is in fact a negative mapping property. Wether instability has to be judged as negative property might depend on several factors. Finally, this question has to be answered with respect to the properties and requirements of the concrete application that makes use of the distributed terminology. Nevertheless, a system that assists an ontology engineer in its work should be able to decide wether or not mappings are stable, and should also, in the case of instability, show with respect to which concepts mappings are instable. In section 7.1 an example is given .

This section ends with a simple proposition about the relation between consistency and stability that can be accepted without further evidence. It has already been argued above that every stable mapping is a consistent mapping.

Proposition 1 *Given $\mathfrak{T}$, if $\mathbb{M}$ is stable with respect to $\mathcal{T}_i$, then $\mathbb{M}$ is consistent with respect to $\mathcal{T}_i$.*

Using the properties of consistency and stability one can make statements about differences between the local and the distributed taxonomy of a terminology. In the next subsection this approach will be extended by comparing the distributed taxonomy to the distributed taxonomy that results from a modified set of bridge rules.

5.3 Invasiveness and Irreducibility

Assume that $\mathcal{M}_{ij}$ - the mapping from $\mathcal{T}_i$ to $\mathcal{T}_j$ - is built up step by step starting with an empty mapping. In the first step bridge rule b_0 is added to $\mathcal{M}_{ij}$, in the second step b_1 and so on. In each step compare the distributed taxonomy of $\mathcal{T}_j$ before adding the bridge rule to the distributed taxonomy of $\mathcal{T}_j$ after adding the bridge rule. Obviously, the distributed taxonomy before the first step - $\mathcal{M}_{ij}$ is still empty - is equal to the local taxonomy. For each step either the taxonomy changes or remains unchanged. If there is at least one modifying step while constructing $\mathcal{M}_{ij}$, the mappings of $\mathfrak{T}$ are instable with respect to $\mathcal{T}_j$.

A statement about the modifiying effect of step k is actually a statement about the relation between $\{b_0, \ldots, b_{k-1}\}$ and b_k in the context of $\mathfrak{T}$. Refrain now from this iterative procedure and simply judge the impact of adding a bridge rule to $\mathcal{M}_{ij}$. It still can be distinguished between bridge rules that change the distributed taxonomy and bridge rules that have no impact on the distributed taxonomy. The following definition introduces the property of invasiveness to distinguish bridge rules according to these considerations.

Definition 6 (Invasiveness) *Given $\mathfrak{T}$, let $\mathfrak{T}'$ be $\mathfrak{T}$ with bridge rule $i : C \xrightarrow{R} j : D$ ($R \in \{\sqsubseteq, \sqsupseteq, \equiv\}$) added to $\mathcal{M}_{ij}$. Then $i : C \xrightarrow{R} j : D$ is invasive with respect to*

$\mathfrak{T}$ iff there exists a pair of concepts $\langle E, F\rangle$ from $\mathcal{T}_j$ such that $\mathfrak{T}' \models j : E \sqsubseteq j : F$ and $\mathfrak{T} \not\models j : E \sqsubseteq j : F$.

What happens if one of the bridge rules in $\mathcal{M}_{ij}$ is removed? Again, it can be distinguished between a modification of the distributed taxonomy and an unmodified taxonomy. If none of the bridge rules in $\mathcal{M}_{ij}$ can be removed without causing a modification, $\mathcal{M}_{ij}$ cannot be reduced to a smaller subset of bridge rules that has the same modifying effect as $\mathcal{M}_{ij}$. The accordant property is called irreducibility and defined as follows.

Definition 7 (Irreducibility) *Given $\mathfrak{T}$, $\mathbb{M}$ is irreducible from $\mathcal{T}_i$ to $\mathcal{T}_j$ iff for every $b \in \mathcal{M}_{ij}$ bridge rule b is invasive with respect to $\mathfrak{T}'$ where $\mathfrak{T}'$ is defined as the distributed terminology $\mathfrak{T}$ with $\mathcal{M}_{ij}$ replaced by $\mathcal{M}_{ij}\backslash\{b\}$. Otherwise $\mathbb{M}$ is reducible from $\mathcal{T}_i$ to $\mathcal{T}_j$. $\mathbb{M}$ is irreducible with respect to $\mathcal{T}_j$ iff for every known source $\mathcal{T}_i$ of $\mathcal{T}_j$ the mappings of $\mathfrak{T}$ are irreducible from $\mathcal{T}_i$ to $\mathcal{T}_j$. Otherwise $\mathbb{M}$ is reducible with respect to $\mathcal{T}_j$.*

Due to the results presented in section 7.1 the irreducible subset of a mapping is almost very small compared to the size of the original mapping. There are two major benefits of a smaller mapping. On the one hand the reasons for instability become obvious in the small, irreducible mapping (compare section 7.1). On the other hand a smaller mapping allows faster reasoning (demonstrated in section 7.2). Aside from any empirical results, with respect to stable mappings the following proposition holds.

Proposition 2 *Given $\mathfrak{T}$, let $\mathbb{M}$ be stable with respect to $\mathcal{T}_j$. If $\mathbb{M}$ is irreducible from $\mathcal{T}_i$ to $\mathcal{T}_j$ then $\mathcal{M}_{ij} \equiv \emptyset$.*

Assume that $\mathbb{M}$ is stable with respect to $\mathcal{T}_j$ and irreducible from $\mathcal{T}_i$ to $\mathcal{T}_j$. Furthermore, let $\mathcal{M}_{ij}$ be a non-empty mapping with $b \in \mathcal{M}_{ij}$. From irreducibility it follows that removing b has effects on the distributed taxonomy of $\mathcal{T}_j$. Thus, the distributed taxonomy differs from the local taxonomy of $\mathcal{T}_j$, and $\mathbb{M}$ therefore cannot be stable with respect to $\mathcal{T}_j$. The assumption has to be rejected and proposition 2 has been proved.

The mapping properties introduced in the last two subsections were built upon satisfiability of concepts respectively upon differences between taxonomies. In both cases mapping properties can be reduced to their effect on relations between concepts. In the following two subsections several properties will be introduced that cannot be described this way. Therefore, attention has to be focused not on the concepts themselves but on their images.

5.4 Extended Terminology and Embedding

The image $C^{i\to j}$ of a concept C from $\mathcal{T}_i$ is a concept in $\mathcal{T}_j$ that fulfills the condition $r_{ij}(C^{\mathcal{I}_i}) \equiv (C^{i\to j})^{\mathcal{I}_j}$ where r is the domain relation connecting elements of the interpretation domains of different terminologies. This is one way to define the notion of an image. Instead of this approach an equivalent but more constructive definition is introduced. The image of a concept can alternatively be defined by extending the distributed terminology $\mathfrak{T}$ in the following way: (1) Add a new concept, the image $C^{i\to j}$ of C, to $\mathcal{T}_j$. (2) Add the equivalence bridge rule $i : C \xrightarrow{\equiv} j : C^{i\to j}$ to $\mathcal{M}_{ij}$. Since $C^{i\to j}$ is linked to C via the equivalence bridge rule, $r_{ij}(C^{\mathcal{I}_i}) \equiv (C^{i\to j})^{\mathcal{I}_j}$ for any model $\mathfrak{I}$ of $\mathfrak{T}$. The following is the formal representation of this constructive definition.

Definition 8 (Extended distributed terminology and image) *Given $\mathfrak{T}$, let $\mathcal{T}'_j$ be $\mathcal{T}_j$ extended with $C^{i\to j}$, where $C^{i\to j}$ is a concept that is not in $\mathcal{T}_j$. Further, let $\mathcal{M}'_{ij}$ be $\mathcal{M}_{ij}$ extended with $i : C \xrightarrow{\equiv} j : C^{i\to j}$. Then concept $j : C^{i\to j}$ is called the image of concept $i : C$ in $\mathcal{T}_j$ and the distributed terminology $\mathfrak{T}^{C^{i\to j}} = \langle\{\mathcal{T}_k\}_{k\neq j,k\in I} \cup \{\mathcal{T}'_j\}, \{\mathcal{M}_{kl}\}_{k\neq i,l\neq j,k\neq l\in I} \cup \mathcal{M}'_{ij}\rangle$ is called $\mathfrak{T}$ extended by $C^{i\to j}$.*

It is now possible to check relations between concepts of different terminologies. This possibility will be exploited in the following to introduce several mapping properties. At first a property is defined that holds if the image $C^{i\to j}$ of a satisfiable concept $i : C$ is distributed unsatisfiable in the extended distributed terminology $\mathfrak{T}^{C^{i\to j}}$. This might occur if there is no unsatisfiable concept in $\mathcal{T}_j$ and even if the mappings are consistent with respect to $\mathcal{T}_j$. Consider, for example, a mapping that consists of the bridge rules $i : C \xrightarrow{\sqsubseteq} j : D$ and $i : C \xrightarrow{\sqsubseteq} j : D'$ with $j : D$ and $j : D'$ being defined as disjoint classes. It follows that $C^{i\to j}$ is distributed unsatisfiable in $\mathfrak{T}^{C^{i\to j}}$.

Definition 9 (Embedding) *Given $\mathfrak{T}$, $\mathbb{M}$ is embedding $i : C$ in $\mathcal{T}_j$ iff $\mathfrak{T} \not\models i : C \sqsubseteq \bot \Rightarrow \mathfrak{T}^{C^{i\to j}} \not\models j : C^{i\to j} \sqsubseteq \bot$. $\mathbb{M}$ is embedding $\mathcal{T}_i$ in $\mathcal{T}_j$ iff for all $i : C$ $\mathbb{M}$ is embedding $i : C$ in $\mathcal{T}_j$. $\mathbb{M}$ is an embedding for $\mathcal{T}_j$ iff for all $\mathcal{T} \in \mathbb{T}$ $\mathbb{M}$ is embedding $\mathcal{T}$ in $\mathcal{T}_j$.*

The experimental results with respect to the property of embedding are similar to the results of checking consistency. In both cases a great number of mappings is judged as defective (see section 7.1).

Notice that this definition does not restrict $\mathcal{T}_i$ to be a known source of $\mathcal{T}_j$. For every terminology $\mathcal{T}_i$ that is not a known source of $\mathcal{T}_j$ and for every concept $i : C$, the extended mapping $\mathcal{M}'_{ij}$ of $\mathfrak{T}^{C^{i\to j}}$ consists of exactly one bridge rule connecting $i : C$ to its image $C^{i\to j}$. Since there is also no relation between $C^{i\to j}$ and any of the

concepts in $\mathcal{T}_j$ the image $C^{i \to j}$ cannot be distributed unsatisfiable. Thus, for every terminology $\mathcal{T}_i$ that is not a known source of $\mathcal{T}_j$ the mappings of $\mathfrak{T}$ are embedding $\mathcal{T}_i$ in $\mathcal{T}_j$.

The property of embedding represents the external counterpart of consistency. Both properties depend on the existence of a distributed unsatisfiable concept respectively image. However, there is no conditional relation that could be expressed by a proposition as one might expect. Embedding does not hold if an image of a satisfiable concept from a different terminology becomes distributed unsatisfiable and inconsistency holds if a satisfiable concept becomes distributed unsatisfiable in its own terminology.

5.5 Entailment and Minimality

The following definitions are centered around the concept of entailment. A bridge rule is entailed by a distributed terminology $\mathfrak{T}$ if the bridge rule does not provide any additional pieces of information that are not explicit or implicit available in $\mathfrak{T}$. Suppose for example that $\mathcal{T}_1$ contains the axiom $Tiger \sqsubseteq Carnivore$ and consider the following bridgerules.

$$1 : Carnivore \xrightarrow{\sqsubseteq} 2 : Animal \tag{7}$$

$$1 : Tiger \xrightarrow{\sqsubseteq} 2 : Animal \tag{8}$$

Obviously, the second bridge rule follows from the first one. The following definition formally introduces the corresponding notion of entailment.

Definition 10 (Entailment of a bridge rule) *Given $\mathfrak{T}$, a bridge rule $i : C \xrightarrow{R} j : D$ with $R \in \{\sqsubseteq, \sqsupseteq, \equiv\}$ is entailed by $\mathfrak{T}$ iff every model $\mathfrak{I}$ of $\mathfrak{T}$ satisfies $i : C \xrightarrow{R} j : D$.*

On the one hand the property of entailment can be used to check wether a bridge rule that is not an element of $\mathfrak{T}$'s mappings can be derived from $\mathfrak{T}$. On the other hand entailment can be used to construct a subset of a mapping $\mathcal{M}_{ij}$ that is strong enough to entail every bridge rule in $\mathcal{M}_{ij}$. Therefore, the related property of minimality has to be defined.

Definition 11 (Minimality) *Given $\mathfrak{T}$, $\mathbb{M}$ is minimal from $\mathcal{T}_i$ to $\mathcal{T}_j$ iff for every $b \in \mathcal{M}_{ij}$ bridge rule b is not entailed by $\mathfrak{T}'$ where $\mathfrak{T}'$ is defined as the distributed terminology $\mathfrak{T}$ with $\mathcal{M}_{ij}$ replaced by $\mathcal{M}_{ij} \backslash \{b\}$. $\mathbb{M}$ is minimal with respect to $\mathcal{T}_j$ iff for every known source $\mathcal{T}_i$ of $\mathcal{T}_j$ the mappings of $\mathfrak{T}$ are minimal from $\mathcal{T}_i$ to $\mathcal{T}_j$.*

Due to the experimental evaluation, automatic generation of mappings produces a great number of bridge rules that are entailed by a smaller minimal subset. This result will be discussed in section 7.1.

	Concept	Image
Satisfiability	Consistency	Embedding
Taxonomies	Stability	-
Modifications of taxonomies	Invasiveness Irreducibility	Entailment Minimality

Table 1: Classification of mapping properties

5.6 Classification of Mapping Properties

The defined properties can be classified with respect to two dimensions. The first dimension is the object of the property. In the case of consistency, stability, invasiveness, and irreducibility it depends on concepts if one of these properties holds, while in the case of embedding, entailment, and minimality images are relevant. The second dimension is the context of the property. Consistency and embedding evaluate concepts respectively images in the context of satisfiability. Invasiveness and entailment evaluate concepts respectively images in the context of mapping modifications and resulting taxonomy modifications. The very same obtains for irreducibility and minimality since these properties are defined in terms of invasiveness and entailment. The classification based on these dimensions is summarized in table 1.

One might wonder why there is an empty cell in the table. The explanation is quite simple. In order to check stability, the local taxonomy is compared to the distributed taxonomy. If there is a concept that has changed its place in the taxonomy of $\mathcal{T}_i$, the mappings are instable with respect to $\mathcal{T}_i$. Such a comparison is not possible for images because it would require that the source terminologies of $\mathcal{T}_i$ could be compared to terminology $\mathcal{T}_i$.

But what about embedding? Why is a comparison possible across the borders of terminologies? This time concepts can be compared to their images, because the empty set $\bot$ is the same in all terminologies. Thus, the unsatisfiability of some concept $i : C$ can be compared to the unsatisfiability of $j : C^{i \to j}$.

This section ends with an important proposition about the relation between invasiveness and entailment that will be used in section 6.3 to optimize the algorithm for computing irreducibility.

Proposition 3 *Given $\mathfrak{T}$, if a bridge rule b is entailed by $\mathfrak{T}$ then b is not invasive with respect to $\mathfrak{T}$.*

It is not very hard to proof this proposition. Let b be a bridge rule $i : C \xrightarrow{R} j : D$ that is invasive with respect to $\mathfrak{T}$. Then there exists a pair of concepts $\langle E, F \rangle$ in $\mathcal{T}_j$

such that $\mathfrak{T}' \models j : E \sqsubseteq F$ and $\mathfrak{T} \not\models j : E \sqsubseteq F$ with $\mathfrak{T}'$ being $\mathfrak{T}$ extended by b. Now let $\mathfrak{I}$ be an interpretation that is a model of $\mathfrak{T}$ but a countermodel of $\mathfrak{T}'$. Since $\mathfrak{T}$ and $\mathfrak{T}'$ differ with respect to $j : E \sqsubseteq F$, such an interpretation exists. Thus, $\mathfrak{I}$ is a modell of $\mathfrak{T}$ that does not satisfy b, and therefore b is not entailed by $\mathfrak{T}$. It can be concluded that every bridge rule that is entailed by $\mathfrak{T}$ is not invasive with respect to $\mathfrak{T}$.

6 Checking Mapping Properties

In the following algorithms for checking the previously defined properties will be stated. Most of these algorithms are straight forward implementations of the correspondent definitions. For the more complicated algorithms a proof of correctness will be sketched. It will also be described how these algorithms have been implemented using the DRAGO system. Finally, the runtime of the algorithms will be discussed with respect to their implementation using DRAGO reasoning services.

6.1 Algorithms

The pseudocode used for describing the algorithms abstracts from many concrete details. A function call like GETALLCONCEPTS($\mathfrak{T}, i$), for example, has two arguments, a distributed terminology $\mathfrak{T}$ and an index $i \in I$. You should notice that in an implementation the index has to be replaced by an address (e.g. a hostame and a portnumber). Before requesting any information, a connection to a reasoning peer has to be established via the specified address. Whereas the distributed terminology $\mathfrak{T}$, the first parameter, never has to be specified. It is determined by the reasoning peer and the fact that this peer is a node in a network of terminologies and mappings. Nevertheless, this simplification makes it easier to focus on the more important aspects of the algorithms and to connect them to the corresponding definitions.

Consistency Algorithm 1 is a straight forward implementation of the definition of consistency. The algorithm iterates over all concepts in $\mathcal{T}_i$ and checks for every concept local and distributed satisfiability.

Stability Checking stability can be reduced to a comparison between the local and the distributed taxonomy of a terminology. Any differences between local and distributed taxonomies always come along with at least one concept having more superclasses in the distributed taxonomy than in the local taxonomy. This principle is used in algorithm 2. The functions GETLOCALSUPERCLASSES($\mathfrak{T}, i, C$) and

Algorithm 1

IsConsistent($\mathfrak{T}, i$)

1: **for all** $C \in$ GetAllConcepts($\mathfrak{T}, i$) **do**
2: **if** $\mathcal{T}_i \not\models C \sqsubseteq \bot$ **and** $\mathfrak{T} \models i : C \sqsubseteq \bot$ **then**
3: **return false**
4: **end if**
5: **end for**
6: **return true**

GetDistributedSuperclasses($\mathfrak{T}, i, C$) are supposed to return the set of all superclasses of concept C with respect to the local respectively the distributed taxonomy of $\mathcal{T}_i$.

Algorithm 2

IsStable($\mathfrak{T}, i$)

1: **for all** $C \in$ GetAllClasses($\mathfrak{T}, i$) **do**
2: $L \leftarrow$ GetLocalSuperclasses($\mathfrak{T}, i, C$)
3: $D \leftarrow$ GetDistributedSuperclasses($\mathfrak{T}, i, C$)
4: **if** $|L| \neq |D|$ **then**
5: **return false**
6: **end if**
7: **end for**
8: **return true**

Embedding The algorithm for checking if the mappings of a distributed terminology are embedding $\mathcal{T}_i$ in $\mathcal{T}_j$ is similar to the algorithm for checking consistency with respect to $\mathcal{T}_j$. In the case of consistency for every concept C local and distributed satisfiability are considered. In the case of embedding it is checked if C is distributed satisfiable in $\mathcal{T}_i$ and if $C^{i \to j}$ is distributed satisfiable in $\mathcal{T}_j$. Algorithm 3 implements this procedure. The more general property of being an embedding for a certain terminology $\mathcal{T}_j$ can be decided by iterating over all known sources $\mathcal{T}_i$ of $\mathcal{T}_j$ each time calling IsEmbedding($\mathfrak{T}, i, j$).

Invasiveness The invasiveness of a bridge rule can be checked by computing the distributed taxonomy before and after adding the bridge rule, as stated in algorithm 4. Notice that the function GetDistributedTaxonomy($\mathfrak{T}, j$) returns the distributed taxonomy of $\mathcal{T}_j$ in an appropriate tree structure that conains full infor-

Algorithm 3

IsEmbedding($\mathfrak{T}, i, j$)

1: **for all** $C \in$ GetAllClasses($\mathfrak{T}, i$) **do**
2: **if** $\mathfrak{T} \not\models i : C \sqsubseteq \bot$ **and** $\mathfrak{T}^{C^{i \to j}} \models j : C^{i \to j} \sqsubseteq \bot$ **then**
3: **return false**
4: **end if**
5: **end for**
6: **return true**

mation about subclasses, equivalent classes, and superclasses for every concept. Thus, the comparison in line four is a complex operation.

Algorithm 4

IsInvasive($\mathfrak{T}, i : C \xrightarrow{R} j : D$)

1: $T \leftarrow$ GetDistributedTaxonomy($\mathfrak{T}, j$)
2: $\mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \cup \{i : C \xrightarrow{R} j : D\}$
3: $T' \leftarrow$ GetDistributedTaxonomy($\mathfrak{T}, j$)
4: **if** $T \neq T'$ **then**
5: **return true**
6: **end if**

Irreducibility Algorithm 5 is computing the irreducible set of bridge rules from $\mathcal{T}_i$ to $\mathcal{T}_j$. Remember that there is no bridge rule in an irreducible mapping that can be removed without changing the distributed taxonomy. Thus, algorithm 5 iterates over all bridge rules and checks if the actual bridge rule b is removable without affecting the distributed taxonomy. If b is removable, the algorithm continues with the reduced mapping. Otherwise b is added again to the mapping. The algorithm terminates if all bridge rules have been processed like this. Then the reduced set of bridge rules $\mathcal{M}_{ij}$ is returned.

Is the resulting mapping - $\mathcal{M}'_{ij}$ in the following argumentation - really the irreducible subset of $\mathcal{M}_{ij}$? Since the distributed taxonomy is neved changed through the whole iteration, $\mathcal{M}'_{ij}$ is definitely a superset of the irreducible set. Otherwise an invasive bridge rule would have been removed. But there still could be a bridge rule b in $\mathcal{M}'_{ij}$ that is not invasive with respect to $\mathcal{M}'_{ij} \backslash \{b\}$. Assume that there is such a bridge rule b. Then there must be a step in the iteration in which b is checked and judged as invasive. Otherwise it would have been removed and would thus not be an element of $\mathcal{M}'_{ij}$. The existence of such a step is equivalent to the existence

Algorithm 5

REDUCEFROMTO($\mathfrak{T}, i, j$)

1: **for all** $b \in \mathcal{M}_{ij}$ **do**
2: $\quad \mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \backslash \{b\}$
3: $\quad$ **if** ISINVASIVE($\mathfrak{T}, b$) **then**
4: $\quad\quad \mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \cup \{b\}$
5: $\quad$ **end if**
6: **end for**
7: **return** $\mathcal{M}_{ij}$

of a set of bridge rules $\mathcal{M}''_{ij}$ with $\mathcal{M}_{ij} \supseteq \mathcal{M}''_{ij} \supseteq \mathcal{M}'_{ij}$ such that b is invasive with respect to $\mathcal{M}''_{ij} \backslash \{b\}$. Compare now $\mathcal{M}'_{ij} \backslash \{b\} \subseteq \mathcal{M}''_{ij} \backslash \{b\} \subseteq \mathcal{M}''_{ij}$ with respect to the corresponding distributed taxonomies. $\mathcal{M}'_{ij} \backslash \{b\}$ and $\mathcal{M}''_{ij}$ result in the same taxonomy, while $\mathcal{M}''_{ij} \backslash \{b\}$ results in a different taxonomy. This would mean that the modifications that have been applied to a taxonomy by adding a set of bridge rules can be undone by adding a further set of bridge rules. But this is definitely not possible. Adding invasive bridge rules is an irreversible operation since every invasive bridge rule is adding further specifications to the distributed taxonomy. Thus, the assumption of $\mathcal{M}'_{ij}$ not being the irreducible subset of $\mathcal{M}_{ij}$ can be rejected. Algorithm 5 correctly computes the irreducible subset of $\mathcal{M}_{ij}$.

Entailment Remember that entailment of a bridge rule b is based on the fact that every model of $\mathfrak{T}$ satisfies b. This cannot be checked directly using standard reasoning methods. The following proposition has to be used.

Proposition 4 (Bridge rule equivalence) *Given $\mathfrak{T}$, for any two concepts $i : C$ and $j : D$ the following equivalences hold.*

$$\begin{aligned}
\mathfrak{T} \models i : C \xrightarrow{\equiv} j : D \quad &\iff \quad \mathfrak{T}^{C^{i \to j}} \models j : C^{i \to j} \equiv D \\
\mathfrak{T} \models i : C \xrightarrow{\sqsubseteq} j : D \quad &\iff \quad \mathfrak{T}^{C^{i \to j}} \models j : C^{i \to j} \sqsubseteq D \\
\mathfrak{T} \models i : C \xrightarrow{\sqsupseteq} j : D \quad &\iff \quad \mathfrak{T}^{C^{i \to j}} \models j : C^{i \to j} \sqsupseteq D
\end{aligned}$$

Now entailment of a bridge rule $i : C \xrightarrow{R} j : D$ can be checked following a very simple procedure. Extend $\mathfrak{T}$ to $\mathfrak{T}^{C^{i \to j}}$ and check relation R between $j : C^{i \to j}$ and $j : D$ in $\mathfrak{T}^{C^{i \to j}}$. Algorithm 6 describes this procedure in detail.

Minimality Having once established a method for checking entailment, an algorithm for computing minimal mappings (algorithm 7) can be stated using an

Algorithm 6

ISENTAILED($\mathfrak{T}, i : C \xrightarrow{R} j : D$)

1: **if** $R = \sqsubseteq$ **then**
2: **return** $\mathfrak{T}^{C^{i\to j}} \models j : C^{i\to j} \sqsubseteq D$
3: **else if** $R = \sqsupseteq$ **then**
4: **return** $\mathfrak{T}^{C^{i\to j}} \models j : C^{i\to j} \sqsupseteq D$
5: **else if** $R = \equiv$ **then**
6: **return** $\mathfrak{T}^{C^{i\to j}} \models j : C^{i\to j} \equiv D$
7: **end if**

iterative procedure that is similar to the algorithm for computing irreducible mappings: For all bridge rules b in $\mathcal{M}_{ij}$ the following operation is applied. Remove b from $\mathcal{M}_{ij}$ and check if b is entailed by the remaining bridge rules. If b is entailed, continue directly with the next bride rule. Otherwise put b back into $\mathcal{M}_{ij}$ and continue with the next bridge rule. Having checked every bridge rule this way, $\mathcal{M}_{ij}$ has become a minimal mapping.

Algorithm 7

MINIMIZEFROMTO($\mathfrak{T}, i, j$)

1: **for all** $b \in \mathcal{M}_{ij}$ **do**
2: $\mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \backslash \{b\}$
3: **if not** ISENTAILED($\mathfrak{T}, b$) **then**
4: $\mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \cup \{b\}$
5: **end if**
6: **end for**
7: **return** $\mathcal{M}_{ij}$

Once again it has to be proved that the resulting $\mathcal{M}_{ij}$ - $\mathcal{M}'_{ij}$ in the folllowing argumentation - is really minimal. On the one hand a bridge rule that cannot be entailed from $\mathcal{M}_{ij}$ or a subset of $\mathcal{M}_{ij}$ is never removed. Thus, the result of the algorithm is definitely a superset of the minimal set. But could there be, on the other hand, an element b in $\mathcal{M}'_{ij}$ that is entailed by $\mathcal{M}'_{ij} \backslash \{b\}$? Such an element does not exist. Since b is entailed by $\mathcal{M}'_{ij} \backslash \{b\}$, it also has to be entailed by any superset of $\mathcal{M}'_{ij} \backslash \{b\}$. Thus, it would have been removed in one of the iterations and it can be concluded that the algorithm is correct.

6.2 Implementation

The algorithms defined in the previous section have been implemented using the DRAGO system described in section 3. The implementation of algorithms 1 and 2 is a straight forward task using standard reasoning services of DRAGO. The same does not hold for the remaining algorithms. Reasoning in the context of an extended terminology $\mathfrak{T}^{C^{i\rightarrow j}}$ requires a mapping with an additional equivalence bridge rule in $\mathcal{M}_{ij}$ and an additional concept $C^{i\rightarrow j}$ in terminology $\mathcal{T}_j$. Minimization and reduction requires the stepwise removal of bridge rules. But the reasoning service interface does not offer any methods for applying changes to the hosted terminology as well as to registered mappings.

Therefore, the following workaround has been implemented. Assume that algorithm 6 is used to check entailment of a bridge rule $i : C \xrightarrow{\sqsubseteq} j : D$ and that $\mathcal{T}_j$ is hosted by DRP A. First of all a second DRP A' is created serving as some kind of temporary clone. The ontology and mapping files registered at A are parsed, extended by image and eqivalence bridge rule, and written to the file system as temporary files. These temporary files are registered at A' and A' is started. The reasoning services of A' can now be requested in order to compute $\mathfrak{T}^{C^{i\rightarrow j}} \models j : C^{i\rightarrow j} \sqsubseteq D$. Afterwards temporary files are deleted and A' is not needed anymore.

The same strategy has been chosen for the other algorithms that require modifications of ontologies or mappings. With respect to irreducibility and minimality the described procedure has to be applied iterative, resulting in numerous IO-operations. This disadvantage hardly affects the overall runtime since the main costs are always related to the reasoning tasks. Nevertheless, in future implementations the functionality of extending (or reducing) ontologies and mappings should better be part of the reasoners functionality. But there is also an advantage of this strategy. The reasoning services of A are not affected by checking mapping properties. Since A's ontology and mappings are not modified, the results of reasoning are still correct. The computational load for checking mapping properties is shifted to the clone of A.

6.3 Comparing Runtimes

The goal of this section is to establish a better understanding of the algorithms and their runtime from a theoretical point of view. Concrete runtime measurements are presented in section 7.1. The reasoning services of DRAGO can be divided into two groups of operations. The first group consists of method calls that force DRAGO to compute the taxonomy of a terminology. This operation is executed whenever the methods GETLOCAL/ DISTRIBUTEDSUPERCLASSES($\mathfrak{T}, i, C$) and

Algorithm	Worst case runtime
(1) IsConsistent($\mathfrak{T}, i$)	$\lvert\mathcal{T}_i\rvert * (\text{Sub}_{L(i)} + \text{Sub}_{D(i)})$
(2) IsStable($\mathfrak{T}, i$)	$\text{Tax}_{L(i)} + \text{Tax}_{D(i)}$
(3) IsEmbedding($\mathfrak{T}, i, j$)	$\lvert\mathcal{T}_i\rvert * (\text{Sub}_{D(i)} + \text{Sub}_{D(j)})$
(4) IsInvasive($\mathfrak{T}, i : C \xrightarrow{R} j : D$)	$\text{Tax}_{D(j)} + \text{Tax}_{D(j)}$
(5) ReduceFromTo($\mathfrak{T}, i, j$)	$(\lvert\mathcal{M}_{ij}\rvert + 1) * \text{Tax}_{D(j)}$
(6) IsEntailed($\mathfrak{T}, i : C \xrightarrow{R} j : D$)	$\text{Sub}_{D(j)}$
(7) MinimizeFromTo($\mathfrak{T}, i, j$)	$\lvert\mathcal{M}_{ij}\rvert * \text{Sub}_{D(j)}$

Table 2: Worst case runtimes of the presented algorithms.

GetLocal/DistributedTaxonomy($\mathfrak{T}, j$) are called. Notice that the results of the classification process are cached. This means that a second method call returns the cached results as far as the same terminology is concerned. The second group consists of all methods that are used to check entailment. Whenever satisfiabilty of a concept or subsumption is checked, the same kind of operation is executed. In this case DRAGO does not build a whole classification. Thus, calling a method of the second group is less expensive than calling a method of the first group.

As explained in section 3, both kinds of operation can be executed in local and distributed mode. Therefore, the runtime of the algorithms can be described with respect to four basic operations.

- $\text{Tax}_{L(j)}$ - computes the local taxonomy of $\mathcal{T}_j$
- $\text{Tax}_{D(j)}$ - computes the distributed taxonomy of $\mathcal{T}_j$
- $\text{Sub}_{L(j)}$ - checks local concept subsumption in $\mathcal{T}_j$
- $\text{Sub}_{D(j)}$ - checks distributed concept subsumption in $\mathcal{T}_j$

These four basic operations have been introduced to enable a comparison between the stated algorithms. This comparison is presented in table 2. In order to abstract from minor differences, an expression like $\text{Tax}_{D(j)}$ denotes operations on $\mathcal{T}_j$ as well as operations on variations of $\mathcal{T}_j$. The extension of $\mathcal{T}_j$ by an image is an example for a such a variation. With respect to runtime estimations these differences can be neglected. For the sake of simplicity $|\mathcal{T}_j|$ is used to refer to the number of concepts in $\mathcal{T}_j$.

Algorithm 5 by far has the worst runtime. The runtime of this algorithm is linear with respect to the number of bridge rules, but for each bridge rule a distributed taxonomy has to be computed.[3] Fortunately, algorithm 5 can be optimized by the

[3]Note that the runtime stated in table 2 is not a correct description of a direct implementation. The

use of proposition 3. Since every entailed bridge rule is non-invasive, the number of checking invasiveness can be reduced. The optimized version is presented in algorithm 8.

Algorithm 8

OPTREDUCEFROMTO($\mathfrak{T}, i, j$)

1: **for all** $b \in \mathcal{M}_{ij}$ **do**
2: $\quad \mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \backslash \{b\}$
3: $\quad$ **if not** ISENTAILED($\mathfrak{T}, b$) **then**
4: $\quad\quad$ **if** ISINVASIVE($\mathfrak{T}, b$) **then**
5: $\quad\quad\quad \mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \cup \{b\}$
6: $\quad\quad$ **end if**
7: $\quad$ **end if**
8: **end for**
9: **return** $\mathcal{M}_{ij}$

If algorithm 8 is applied to a minimal mapping, a worse runtime compared to algorithm 5 can be expected. In this case an unnecessary check of entailment is executed for every bridge rule. But for mappings that have a relatively small minimal subset of bridge rules algorithm 8 is definitely a better choice. This will be proved in the context of the experimental evaluation.

Part III

7 Experimental Evaluation

The ontologies used in following experiments origin from the OntoFarm initiative, described in [10].[4] The OntoFarm ontologies have been created to provide a set of rich ontologies that on the one hand describe a similar domain and on the other reflect different conceptualisations of this domain. Therefore, the OntoFarm ontologies are well-suited for creating non-trivial semantic mappings.

All ontologies are about the domain of conference organisation. They were developed based upon (1) actual conferences and corresponding web pages, (2) software tools for conference organisation support, and (3) experience of people with personal participation in conference organisation. Six ontologies of the Onto-

formula in the table describes the runtime of a more efficient algorithm that computes the distributed taxonomy of the initial situation only once and compares it to the taxonomy of every iteration.

[4] The ontology files and additional informations are available at http://nb.vse.cz/ svabo/oaei2006/.

Ontologie	Based upon	Classes	Properties
$\mathcal{T}_{CRS}$	Tool	14	17
$\mathcal{T}_{PCS}$	Tool	23	38
$\mathcal{T}_{CMT}$	Tool	36	59
$\mathcal{T}_{CONFTOOL}$	Tool	38	36
$\mathcal{T}_{SIGKDD}$	Web	49	28
$\mathcal{T}_{EKAW}$	Experience	77	33

Table 3: Ontologies chosen from the OntoFarm collection.

Farm collection have been chosen for empirical evaluation. These ontologies are described in table 3. For each pair of different test ontologies $\langle \mathcal{T}_i, \mathcal{T}_j \rangle$ the CtxMatch matching tool[5] has been used to create a mapping from $\mathcal{T}_i$ to $\mathcal{T}_j$. The algorithm implemented in CtxMatch is an algorithm for discovering semantic mappings across hierarchical classifications (compare [12]).

The chosen OntoFarm ontologies and the generated mappings can be used to construct different kinds of distributed terminologies. In section 7.1 distributed terminologies are built of two ontologies $\mathcal{T}_i$ and $\mathcal{T}_j$ connected via one mapping $\mathcal{M}_{ij}$. In this context $\mathcal{T}_i$ will be called source terminology and $\mathcal{T}_j$ will be called target terminology. In section 7.2 more complex distributed terminologies with multiple source terminologies will be constructed.

7.1 Mapping Diagnosis

An overview of the results and the runtime of the experiments discussed below is presented in table 4. Each cell in the table represents a distributed source-target terminology. Each row corresponds to the set of distributed terminologies that share the same source terminology, while the distributed terminologies belonging to the same column share the same target terminology. The second cell in the first row, for example, refers to the distributed terminology that consistent of $\mathcal{T}_{CRS}$ (source terminoloy) and $\mathcal{T}_{PCS}$ (target terminology). This distributed terminology will be called $\mathfrak{T}_{CRS-PCS}$. The same manner of speaking obtains for the other cells respectively distributed terminologies.[6]

In the first line of each cell the size of mappings is noted in number of bridge rules. The results for inconsistency, instability, and embedding can be found in

[5] CtxMatch is available at http://dit.unitn.it/ zanobini/downloads.html.

[6] Some entries in the table are empty. The corresponding computations have been stopped after an hour since no progress could be detected. All tests have been realised with 512Kb ram on a Pentium M 1400Mhz. For a detailed presentation of the results summarized in table 4 see appendix A.

	$\mathcal{T}_{CRS}$	$\mathcal{T}_{PCS}$	$\mathcal{T}_{CMT}$	$\mathcal{T}_{CONFTOOL}$	$\mathcal{T}_{SIGKDD}$	$\mathcal{T}_{EKAW}$
$\mathcal{T}_{CRS}$		38 0 (1.9) 2 (3.2) 0 (16.9) 45% (47.4) 8% (157.4)	71 2 (0.7) 2 (2.3) 0 (16.3) 30% (64.9) 6% (163.5)	80 27 (0.5) 27 (1.1) 6 (17.2) 12% (101.2) 6% (150.6)	57 0 (0.8) 10 (9.5) 0 (15.7) 40% (64.5) 21% (409.3)	219 29 (1.3) 64 (7.9) 5 (19.3) 9% (282.5) 6% (628.9)
$\mathcal{T}_{PCS}$	38 0 (0.3) 0 (1.3) 3 (26.5) 45% (42.7) 0% (89.6)		89 0 (0.8) 2 (3.3) 0 (27.2) 21% (86.9) 3% (232.7)	45 5 (0.6) 5 (4.0) 5 (26.9) 40% (54.3) 18% (199.2)	56 0 (0.8) 7 (8.1) 0 (25.8) 36% (65.3) 14% (390.4)	126 4 (1.2) 22 (12.6) 0 (28.9) 18% (165.7) 11% (685.2)
$\mathcal{T}_{CMT}$	71 1 (0.3) 1 (0.7) 2 (32.5) 28% (67.3) 4% (125.2)	92 0 (2.1) 6 (5.2)		62 3 (0.5) 3 (4.0) 1 (32.8) 27% (59.3) 10% (173.3)	66 0 (0.8) 7 (8.4) 0 (32.9) 24% (72.1) 14% (296.5)	138 4 (1.5) 22 (14.1) 1 (36.3) 16% (156.9) 8% (706.4)
$\mathcal{T}_{CONFTOOL}$	80 10 (0.2) 10 (0.2) 15 (43.1) 19% (93.2) 5% (124.9)	45 0 (0.4) 0 (1.6) 0 (42.0) 40% (51.1) 0% (127.6)	62 0 (0.4) 0 (1.9) 0 (42.1) 29% (55.9) 0% (131.6)		75 0 (0.6) 7 (8.3) 0 (42.5) 43% (87.5) 11% (393.5)	236 29 (1.6) 57 (9.6) 14 (53.7) 13% (321.2) 6% (516.2)
$\mathcal{T}_{SIGKDD}$	57 1 (0.3) 1 (0.7) 4 (56.6) 40% (70.3) 4% (121.9)	56 0 (0.4) 4 (1.9) 0 (55.1) 36% (79.6) 7% (119.1)	66 0 (0.5) 0 (2.2) 0 (55.7) 24% (76.2) 0% (158.3)	72 3 (0.5) 3 (4.0) 7 (59.1) 42% (83.6) 7% (252.7)		132 4 (2.4) 22 (20.2) 4 (67.5) 24% (150.2) 8% (772.5)
$\mathcal{T}_{EKAW}$	221 10 (0.5) 10 (0.2) 53 (99.3) 7% (230.2) 2% (284.2)	126	134 0 (1.4) 0 (3.4) 4 (94.4) 17% (126.9) 0% (322.1)	238 27 (1.3) 27 (1.3) 45 (112.5) 8% (279.9) 3% (368.4)	131 0 (1.9) 7 (10.4) 0 (97.8) 26% (141.8) 6% (601.4)	

Table 4: Number of bridge rules; number of inconsistent, instable, and not embedded concepts; reduction rate of minimality and irreducibility with respect to number of bridge rules.

the next three lines. Consider the cell corresponding to $\mathfrak{T}_{CRS-PCS}$ again. The second line means that there is no concept C in $\mathcal{T}_{PCS}$ such that the mapping of $\mathfrak{T}_{CRS-PCS}$ is inconsistent with respect to C. The subsequent value in parenthesis refers to the runtime in seconds. In this case computing the set of inconsistent bridge rules took 1.9 seconds. The same obtains for the third line in each cell with respect to instability, and for the fourth line with respect to embedding.

Counting the cells with one or more inconsistent concepts results in 15 of 29 distributed terminologies with an inconsistent mapping. This means that approximately half of the automatically generated mappings are defect with respect to mapping consistency. Thus, the formal property of mapping consistency can be successful used for detecting defective mappings. The runtimes for checking consistency ranges between 0.3 and 2.4 seconds.

Besides consistency, the property of embedding provides a second possibility to sort out defective mappings. 15 of 28 mappings are not embedding their source terminology into their target ontology. Combining the results for both properties only 11 mappings are free from formal errors with respect to consistency or embedding. Remember that consistency and embedding are hard criteria for the evaluation of mappings. This means that about 60% of the generated mappings definitely have to be judged as incorrect for formal reasons.

Can the property of instability also be used in the context of mapping diagnosis? In the column for target terminology $\mathcal{T}_{SIGKDD}$ there are three distributed terminologies (source terminologies $\mathcal{T}_{PCS}$, $\mathcal{T}_{CMT}$, and $\mathcal{T}_{CONFTOOL}$) that have instable but consistent mappings. For all of them there are seven instable concepts, and these concepts, listed in the following enumeration, are the same for all of them.

- $Registration\_SIGMOD\_Member$
- $Registration\_Student$
- $Registration\_NonMember$
- $Registration\_SIGKDD\_Member$
- $Deadline\_Author\_notification$
- $Best\_Student\_Paper\_Award$
- $Conference\_hall$

By comparing the local taxonomy of $\mathcal{T}_{SIGKDD}$ to the distributed taxonomy of $\mathcal{T}_{SIGKDD}$ it shows that for all of the three distributed terminologies the listed concepts have become subclasses of $Person$. Obviously, this cannot be correct. The listed concepts are subclasses of registration fees, deadlines, and places. Thus, by manually checking instable concepts, further defective mappings can be detected.

The fifth and the sixth line present the reduction rate that results from computing the minimal respectively the irreducible mapping from source to target termi-

nology. Notice that the size of the minimal mapping is between 10% and 45% of the original size. This means that a significant part of the automatically generated bridge rules does not yield any new information and is thus redundant. The irreducible subset of bridge rules is even smaller. Notice also that, in accordance to proposition 2, the irreducible set of bridge rules is empty if the original mapping is stable. Figure 3 and 4 present a visualization of an original mapping and the corresponding minimal mapping.[7] Comparing both figures gives a good impression of minimization effects.

A more compact mapping is an advantage in the context of mapping debugging. Consider for example the irreducible mapping of $\mathfrak{T}_{PCS-SIGKDD}$. This mapping consists of the following eight bridge rules.

- $PCS : PERSON \xrightarrow{\equiv} SIGKDD : Person$
- $PCS : PERSON \xrightarrow{\sqsupseteq} SIGKDD : Registration\_SIGMOD\_Member$
- $PCS : PERSON \xrightarrow{\sqsupseteq} SIGKDD : Registration\_Student$
- $PCS : PERSON \xrightarrow{\sqsupseteq} SIGKDD : Registration\_NonMember$
- $PCS : PERSON \xrightarrow{\sqsupseteq} SIGKDD : Registration\_SIGKDD\_Member$
- $PCS : PERSON \xrightarrow{\sqsupseteq} SIGKDD : Deadline\_Author\_notification$
- $PCS : PERSON \xrightarrow{\sqsupseteq} SIGKDD : Best\_Student\_Paper\_Award$
- $PCS : PERSON \xrightarrow{\sqsupseteq} SIGKDD : Conference\_hall$

These bridge rules are exactly the bridge rules that force the concepts listed above to become subclasses of $SIGKDD : Person$. For each instable concept there is a pair of bridge rules, that consists of the first bridge rule and one of the remaining bridge rules, such that $SIGKDD : Person$ subsumes this concept even if only these two bridge rules would be used in distributed reasoning. But since the first bridge rule expresses a correct semantic relation, the remaining bridge rules have to be incorrect. Therefore, the property of irreducibilty can be used to focus on a small subset of bridge rules that results in the unwanted effect of instability.

Meilicke, Stuckenschmidt and Tamilin have introduced the notion of an irreducible conflict set (see [6]), that is closely related to the notion of irreducibility. The authors have shown that automated mapping repairing can be realized by stepwise removing elements from irreducible conflict sets. By applying this procedure inconsistent mappings become consistent. There is some evidence that a similar strategy can be applied with respect to certain subsets of irreducible sets to stabilize a mapping.

Remember that there are two alternatives for computing irreducibility, the di-

[7] The tool that was used to create these visualizations has been developed by Andrei Tamilin and is not part of the authors work.

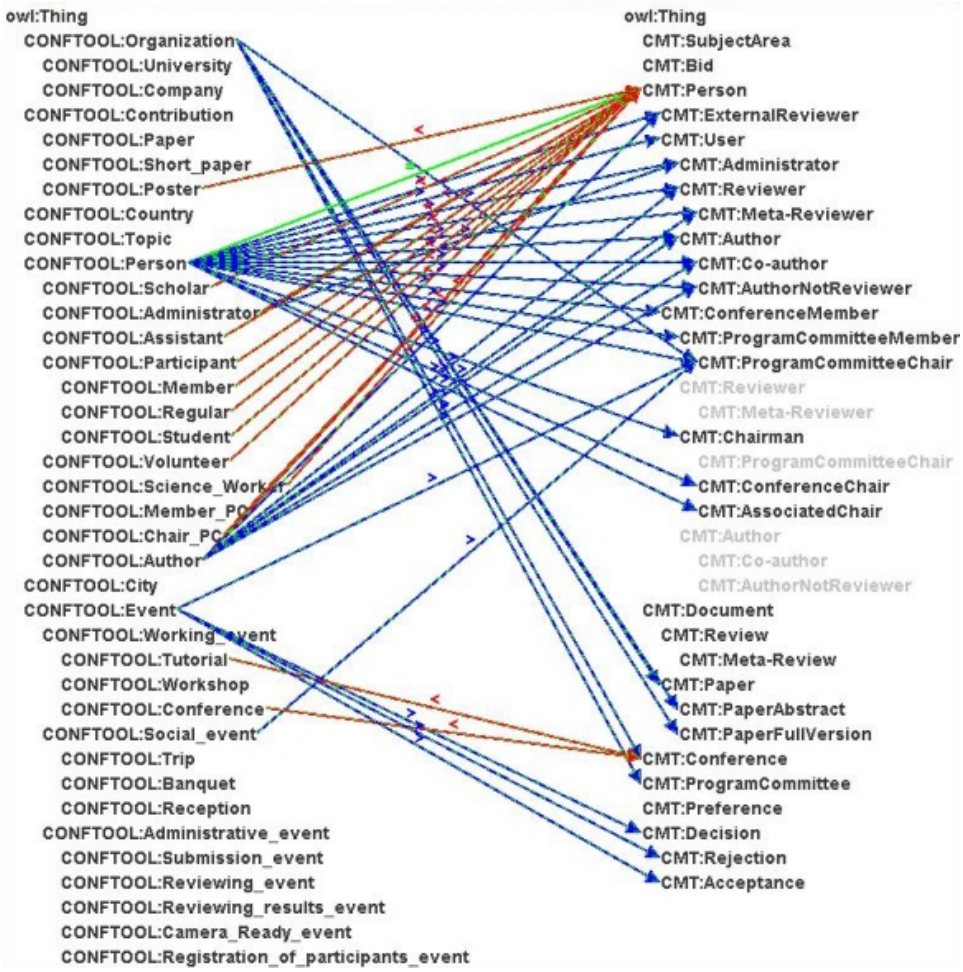

Figure 3: Mapping from $\mathcal{T}_{CONFTOOL}$ to $\mathcal{T}_{CMT}$ before minimization. Arrows captioned with $>$ indicate onto bridge rules, arrows captioned with $<$ indicate into bridge rules, and arrows captioned with = indicate equivalence bridge rules. Multiple occurences of the same concept (concept has different superclasses) are written in grey font. Arrows are only drawn to the first occurence of the concept.

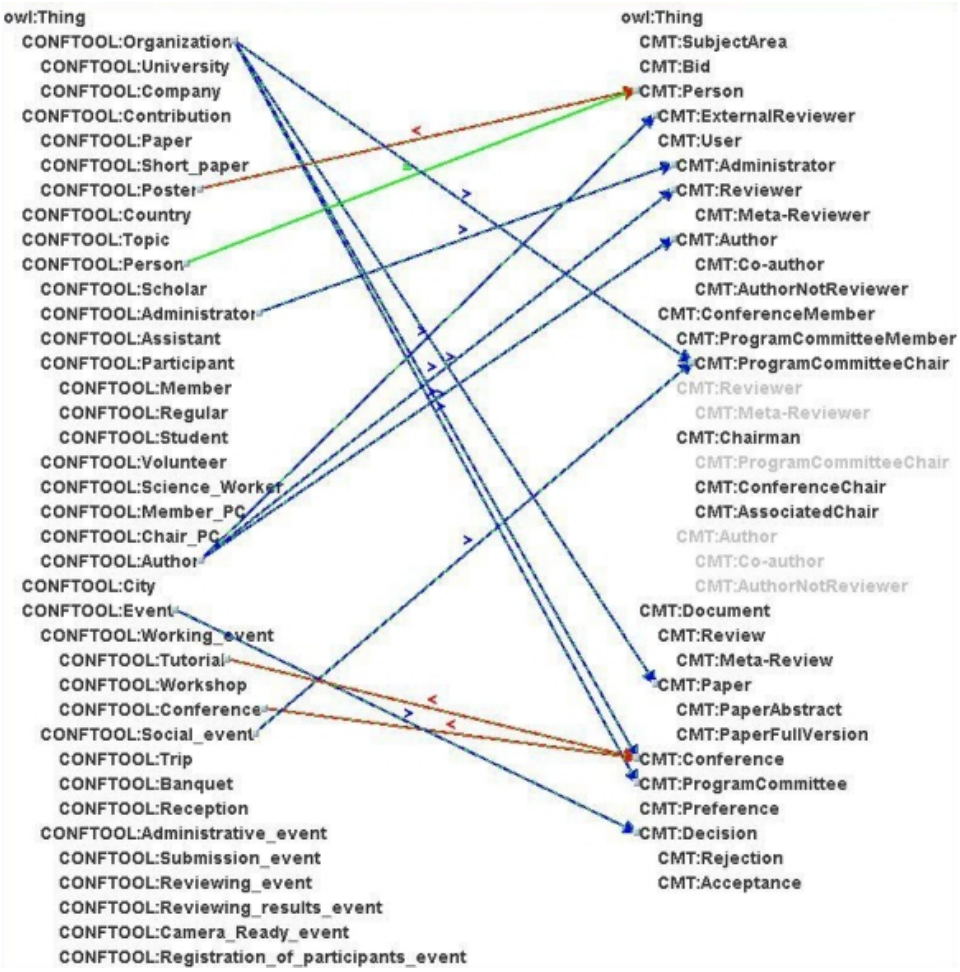

Figure 4: Mapping from $\mathcal{T}_{CONFTOOL}$ to $\mathcal{T}_{CMT}$ after minimization.

rect application of invasiveness (algorithm 5) and the optimized variant (algorithm 8). The runtimes stated in table 4 are based on applying algorithm 8. Additional experiments have shown that the optimized variant is up to five times faster.[8] Nevertheless, in some cases the direct algorithm is as fast as or even slightly faster than its optimized variant. This depends on the number of entailed bridge rules. If the irreducible mapping, for example, is computed based on a minimal mapping, the direct approach will always be faster.

7.2 Reasoning with Minimal and Irreducible Mappings

In this section the properties of minimality and irreducibilty will be discussed with respect to their effects on the runtime of typical reasoning requests. The main idea of this section is quite simple. A mapping with less bridge rules allows faster reasoning since, in the context of the distributed tableau algorithm, less branches can cross the border of a local terminology. Thereby cost-intensive distributed reasoning is reduced. Before the emprical verification of this assertion can be discussed, a few theoretical observations have to be made.

For the following considerations let $\mathfrak{T}_{min}$ and $\mathfrak{T}_{red}$ denote a modified distributed terminology $\mathfrak{T}$ with original mappings replaced by minimal respectively irreducible mappings. Remember that minimality is based on the notion of entailment. A bridge rule b is entailed if every model of the distributed terminology also satisfies b. Thus, any model for $\mathfrak{T}_{min}$ is also a model for $\mathfrak{T}$ and vice versa. The same does not hold with repect to $\mathfrak{T}_{red}$. Working with $\mathfrak{T}_{red}$ only guarantees that the distributed taxonomies of $\mathfrak{T}_{red}$'s terminologies are equal to the distributed taxonomies of $\mathfrak{T}$'s terminologies. Therefore, the relation between $\mathfrak{T}$ and $\mathfrak{T}_{red}$ could be denoted as weak equivalence while the relation between $\mathfrak{T}$ and $\mathfrak{T}_{min}$ could be denoted as (strong) equivalence.

Reasoning in $\mathfrak{T}_{min}$ always yields the same results as reasoning in $\mathfrak{T}$. Thus, for any kind of application a distributed terminology $\mathfrak{T}$ can be replaced by its minimal equivalent $\mathfrak{T}_{min}$. But it depends on the kind of reasoning requests if $\mathfrak{T}$ can also be replaced by $\mathfrak{T}_{red}$.[9] Such a replacement is not always possible and it has to be decided in consideration of certain aspects of the application. This difference between minimal and irreducible mappings has to be comprehended before empirical results can be discussed.

Three different distributed terminologies have been chosen as testcases. Each distributed terminology consists of one target terminology and two source terminologies. $\mathcal{T}_{CRS}$, $\mathcal{T}_{CMT}$ and $\mathcal{T}_{SIGKDD}$ have been chosen as building blocks

[8] See appendix A for detailed runtime measurements.

[9] The actual version of the DRAGO system only offers reasoning services that yield the same results for $\mathfrak{T}_{red}$ and $\mathfrak{T}$.

of the distributed terminologies refered to as $\mathfrak{T}_{CRS+}$, $\mathfrak{T}_{CMT+}$, and $\mathfrak{T}_{SIGKDD+}$. $\mathfrak{T}_{CRS+}$, for example, consists of target terminology $\mathcal{T}_{CRS}$ and source terminologies $\mathcal{T}_{CMT}$ and $\mathcal{T}_{SIGKDD}$. An equivalent structure has been chosen for $\mathfrak{T}_{CMT+}$ and $\mathfrak{T}_{SIGKDD+}$. The mappings connecting the sources with the target are the mappings that have been created by the use of CtxMatch. In the following these mappings will be called the original mappings. In section 7.1 corresponding minimal and irreducible mappings have been computed. These mappings will now be compared with respect to their influence on the runtime of certain reasoning tasks.

The minimal and irreducible mappings from section 7.1 have been computed with respect to a simple distributed terminology consisting of just one source and one target connected via one mapping. A minimal respectively irreducible mapping computed in such a context will be called a pairwise computed mapping. In the context of a more complex distributed terminology, like introduced in this section, pairwise computed mappings are not necessarily minimal or irreducible. Consider the following example in the context of $\mathfrak{T}_{CRS+}$. Assume that the same modification in the distributed taxonomy of $\mathcal{T}_{CRS}$ is caused by a subset $\mathcal{M}'$ of $\mathcal{M}_{CMT-CRS}$ as well as by $\mathcal{M}_{SIGKDD-CRS}$. If first of all the irreducible set of bridge rules corresponding to $\mathcal{M}_{CMT-CRS}$ is computed, the bridge rules in $\mathcal{M}'$ will not be contained in the resulting irreducible set. These bridge rules can be removed without affecting the distributed taxonomy of $\mathcal{T}_{CRS}$ since $\mathcal{M}_{SIGKDD-CRS}$ is strong enough to force the modification mentioned above upon the taxonomy of $\mathcal{T}_{CRS}$. Nevertheless, $\mathcal{M}'$ could be a subset of the pairwise computed irreducible mapping because, restricted to $\mathcal{M}_{CMT-CRS}$, it might be the only causer of the modification.

So why not compute the minimal respectively irreducible subsets in the context of the more complex distributed terminologies? There is a big advantage in using pairwise computed mappings. If one of the reasoning peers hosting a source terminology is not availabe negative effects should be as small as possible. This is guaranteed when using pairwise computed mappings. The assistance of other mappings is not necessary since all of the invasive power has been preserved in the pairwise computed irreducible mapping. Using irreducible mappings that have been computed in a multi source environment is also problematic when new sources are added. To ensure that the mappings are still in a strict sense irreducible a recomputation has to take place in such a situation.

Table 5 comprises the results of reasoning with original and pairwise computed minimal respectively irreducible mappings. Two kinds of operations have been chosen as examples for standard reasoning tasks. The first task is the computation of a distributed taxonomy. The second task is the computation of the set of all mapping inconsistent concepts (see algorithm 1). Computing mapping inconsistency is no standard DRAGO operation. Nevertheless, it yields a good measure

Distributed terminology and mappings	$\mathfrak{T}_{CRS+}$			$\mathfrak{T}_{CMT+}$			$\mathfrak{T}_{SIGKDD+}$		
	irreducible	minimal	original	irreducible	minimal	original	irreducible	minimal	original
Taxonomy (Order 1)	2.22	3.81	4.55	6.49	7.92	9.22	30.71	37.11	39.53
Taxonomy (Order 2)	1.95	3.38	3.93	6.35	7.91	8.78	31.15	38.51	40.03
Consistency (Order 1)	1.19	2.21	2.73	1.36	2.09	2.51	2.40	3.09	3.83
Consistency (Order 2)	0.62	1.30	1.63	1.06	1.72	2.34	3.20	4.22	5.22

Table 5: Runtime comparison using original, minimal and irreducible mapping.

for checking multiple times distributed satisfiability. For both operations table 5 states the runtime in seconds.

There is also a distinction between two different orders. Remember that every terminology $\mathcal{T}_j$ is hosted by a reasoning peer. Another terminlogy $\mathcal{T}_i$ becomes known to a reasoning peer if a mapping $\mathcal{M}_{ij}$ is registered at the peer hosting $\mathcal{T}_j$. A reasoning peer stores mappings in the order of their registration. Whenever requests are propagated to known terminologies this order is maintained. Therefore, different orders of mapping-registration can have effects on the runtime of reasoning.[10]

First consider the runtimes for computing the distributed taxonomy. For all testcases the irreducible mapping performs better than the minimal mapping, which again performs better than the original mapping. The runtimes using the irreducible mapping range between 50% and 80% of the runtimes using the original mapping. The profit of minimization is much smaller. Between 85% and 95% of the original runtimes are reached. With respect to mapping consistency the effects of minimality and irreducibility are stronger. Using the irreducible mapping results in 40% to 65% of the original runtimes. Minimization optimizes the runtime up to 75% and even in the worst case the minimal mapping is still 19% faster than the original mapping.

These results show that the properties of minimality and irreducibility can be used to improve the performance of standard reasoning tasks. But do the demonstrated improvements justify a cost-intensive computation of minimal or irreducible mappings? Consider the following example. Suppose that there is a complex distributed peer network structured as tree with n levels. In such a situation a coeffi-

[10]Order 1: $\mathfrak{T}_{CRS}$ uses $\mathcal{M}_{CMT-CRS}$ before $\mathcal{M}_{SIGKDD-CRS}$, $\mathfrak{T}_{CMT}$ uses $\mathcal{M}_{CRS-CMT}$ before $\mathcal{M}_{SIGKDD-CMT}$, $\mathfrak{T}_{SIGKDD}$ uses $\mathcal{M}_{CRS-SIGKDD}$ before $\mathcal{M}_{CMT-SIGKDD}$. Order 2 is the reversion of order 1.

cient like 0.5 soon becomes interesting with growing n. It can be concluded wether to use or not to use minimal or irreducible mappings depends among other factors also on the strucure of the distributed terminology.

Part IV

8 Summary and Discussion

Both theoretical and experimental investigations have been described in part II and part III of this thesis. The results of these investigations and the contribution to the research questions stated in the first section can be summarized as follows.

Theoretical framework Besides restating some of the definitions suggested in [9], further mapping properties have been introduced. These are the properties of stability, invasiveness, and irreducibility. Several propositions have been stated to describe dependencies between mapping properties. These propositions are not merely of theoretical interest. Realizing that every entailed bridgerule is also a non-invasive bridge rule (see proposition 3), has been the key for the optimized version of the algorithm for computing irreducible sets (algorithm 8). The relation between stability and irreducibility (see proposition 2) enables one to check if the irreducible set of bridge rules is empty without cost-intensive computation of this set. The understanding of dependencies and differences between mapping properties results in the classification presented in table 1.

Algorithms For every property a corresponding algorithm has been stated as well as implemented as DRAGO extension. This implementation has been used in the scope of the experimental evaluation and is delivered as part of this thesis.[11] The results in table 4 show that the runtime performance of the algorithms is acceptable as far as concerned with operations of preprocessing. Thus, this thesis yields both an abstract description and a concrete implementation of efficient algorithms for deciding mapping properties.

Mapping Diagnosis Based on experiments with a set of automatically generated mappings, it has been shown that the properties of inconistency and embedding can be used to detect defective mappings. This is an important feature in the context of automated mapping generation. It has also been demonstrated that the notions

[11] See appendix A for further information.

of stability and irreducibility constitute an assistance in detecting subsets of bridge rules that are causing defects in a mapping. It has already been argued by Meilicke, Stuckenschmidt and Tamilin that a variant of this approach can be extended towards automated mapping debugging (compare [6]).

Runtime The last topic of research is concerned with minimal and irreducible mappings and their effects on runtime performance. Performance improvements up to a factor of 0.4 (irreducibility) respectively 0.75 (minimality) have been measured. There is some evidence that in complex distributed terminololgies even small improvements will result in significant benefits.

There are also a few points of criticism worth mentioning. One should notice that the stated propositions do not exhaustively describe the theoretical dependencies between the introduced mapping properties. Remember, for example, that there is more than just one irreducible or minimal set of bridge rules corresponding to a particular mapping. But there is some evidence that for stable mappings the minimal set is uniquely determined. This has to be proved, and there might be further interesting dependencies that should be detected and proved. Nevertheless, this thesis yields a compact and coherent general view of the theoretical framework.

Most of the algorithms stated in section 6.1 are direct applications of the corresponding definitions. Thus, there are possibly better algorithmic solutions. A need for omptimization is also suggested by the results of the empirical evaluation. Especially the runtimes of computing minimality and irreducibility are very high. This will become problematic with growing size of terminologies and mappings. Thus, further research on optimizing algorithms might be necesarry. But, remember that runtimes being problematic for standard reasoning request are acceptable in the stage of preprocessing.

On might criticise that all of the mappings used as testcases were produced with the same matching tool. Thus, the results collected in table 4 cannot be generalized. On the one hand this point of criticism is applicable. Further investigations will have to show if similar results are obtained by the use of other matching tools. On the other hand a matching tool that guarantees the avoidance of inconsistent and non-embedding mappings will implicitly make use of the introduced properties. The existence of such a tool is not yet known to the author.

References

[1] GRIGORIS ANTONIUS AND FRANK VAN HARMELEN. *Web Ontology Language: OWL*. In: Handbook on Ontologies. Cambridge University Press, 2003.

[2] FRANZ BAADER, IAN HORROCKS, AND ULRIKE SATTLER. *Basic Description Logic*. In: Handbook on Ontologies. Cambridge University Press, 2003.

[3] FRANZ BAADER AND WERNER NUTT. *Basic Description Logic*. In: The Description Logic Handbook - Theory, Implementation and Applications. Cambridge University Press, 2003.

[4] P. BOUQUET, F. GIUNCHIGLIA, F. VAN HARMELEN, L. SERAFINI, AND H. STUCKENSCHMIDT. *C-owl: Contextualizing ontologies*. In: Proceedings of the 2nd International Semantic Web Conference ISWC'03, Lecture Notes in Computer Science, Sanibal Island, Florida. Springer Verlag, 2003.

[5] ALEX BORGIDA AND LUCIANO SERAFINI. *Distributed Description Logics: Assimilating Information from Peer Sources*. In: Journal of Data Semantics, 2003.

[6] CHRISTIAN MEILICKE, HEINER STUCKENSCHMIDT, AND ANDREI TAMILIN. *Improving Automatically Created Mappings using Logical Reasonings*. Submitted to the ISWC'06 workshop on Ontology Matching, Athens, Georgia, USA, 2006.

[7] LUCIANO SERAFINI AND ANDREI TAMILIN. *DRAGO: Distributed reasoning architecture for the semantic web*. In: Proceedings of the Second European Semantic Web Conference (ESWC05). Springer-Verlag, 2005.

[8] LUCIANO SERAFINI AND ANDREI TAMILIN. *Local tableaux for reasoning in distributed description logics*. In: Proceedings of the 2004 Int. Workshop on Description Logics (DL2004), CEUR-WS, 2004.

[9] HEINER STUCKENSCHMIDT, LUCIANO SERAFINI, AND HOLGER WACHE. *Reasoning about Ontology Mappings*. In: Proceedings of the ECAI-06 Workshop on Contextual Representation and Reasoning, 2006.

[10] ONDREJ SVAB, VOJTECH SVATEK, PETR BERKA, DUSAN RAK, AND PETR TOMASEK. *OntoFarm: Towards an Experimental Collection of Parallel Ontologies*. In: Poster Processings of the International Semantic Web Conference 2005, 2005.

[11] ALEX BORGIDA AND LUCIANO SERAFINI. *Distributed Description Logics - Preliminary Investigations.* In: Proceedings of the 2002 Int. Workshop on Description Logics, 2002.

[12] PAOLO BOUQUET, LUCIANO SERAFINI, AND STEFANO ZANOBINI. *Semantic coordination: a new approach and an application.* In: Proceedings of the Second Internatinal Semantic Web Conference, volume 2870 of Lecture Notes in Computer Science. Springer Verlag, 2003.

Appendix

A Program Code and Additional Experimental Results

The source code, a documentation, some usage examples, and additional testresults can be found on the compact disc delivered as part of this thesis. The program has been written in java 1.5, is realized as package de.unima.mapping.ram and delivered as jar-file. It has been tested and developed on a windows system. There are no experiences available using other operating systems. To run the examples copy the directory `ram\` from compact disc to partition `C` on your hard drive.[12] The content of the subdirectories is described in the following enumeration.

- `C:\ram\distribution` - Contains jar-files of the ram package, jar-files of the DRAGO system, all further necessary libraries, and some usage examples. [13]
- `C:\ram\documentation` - Contains a documentation of the user interface (the classes RAM and RAMException) provided by the ram package.
- `C:\ram\sourcecode` - Contains the source code of the ram package.
- `C:\ram\terminologies` - Contains ontologies chosen from the OntoFarm collection and generated mappings. All mappings are also available as minimal and irreducible mappings.
- `C:\ram\testresults` - Contains additional experimental results. Open the contents of this directory in your browser for detailed informations.

The content of the file `C:\ram\distribution\instructions.txt` explains in detail how to compile and run the usage examples.

[12] It is important to maintain the paths used in this description. Otherwise modifications of the usage examples and/or modifications of the mapping files are necessary.

[13] Notice that slight modifications have been applied to the DRAGO system. Therefore, the delivered jar files have to be used instead of the original versions of the DRAGO system.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Bensheim, den 31.08.2006 Unterschrift