Reasoning Support for Mapping Revision

Christian Meilicke[1], Heiner Stuckenschmidt[1] and Andrei Tamilin[2]

[1] Knowledge Representation and Knowledge Management Group,
Computer Science Institute, University of Mannheim,
A5, 6 68159 Mannheim, Germany

[2] Data and Knowledge Management Group,
Foundation Bruno Kessler - IRST
Via Sommarive 18, 38100 Povo di Trento, Italy

April 14, 2008

Abstract

Finding correct semantic correspondences between heterogeneous ontologies is one of the most challenging problems in the area of semantic web technologies. As manually constructing such mappings is not feasible in realistic scenarios, a number of automatic matching tools have been developed that propose mappings based on general heuristics. As these heuristics often produce incorrect results, a manual revision is inevitable in order to guarantee the quality of generated mappings. Experiences with benchmarking matching systems revealed that the manual revision of mappings is still a very difficult problem because it has to take the semantics of the ontologies as well as interactions between mappings into account. In this paper, we propose methods for supporting human experts in the task of revising automatically created mappings. In particular, we present non-standard reasoning methods for detecting and propagating implications of expert decisions on the correctness of a mapping.

Keywords: Ontologies, Ontology Mappings, Mapping Revision, Distributed Description Logics

1 Introduction and Motivation

The integration of information from heterogeneous sources is one of the major challenges of modern information technology. Researchers from different areas including databases, knowledge representation and more recently in semantic web technologies have addressed this problem. Ontologies have been identified as a key technology for resolving semantic heterogeneity by providing common terms as well as formal specifications of their intended meaning in some logic. In large distributed environments with a high number of different information sources, however, it is unlikely that people

will agree on a single ontology as the basis for integrating information. Here, we often face a situation where multiple ontologies describing the very same domain co-exist. In such a situation, we first have to integrate the different ontologies before they can serve as a basis for integrating information.

A common way of integrating different ontologies describing the same or largely overlapping domains is to use formal representations of semantic correspondences between their concepts and relations - also referred to as 'ontology mappings'. Manual approaches for identifying semantic correspondences are often not feasible since real world ontologies, for example in the medical domain, often contain several thousand concepts. As a response to this problem, a number of automatic and semi-automatic tools for generating hypotheses about semantic correspondences have been developed (see [5] for an overview). The results of these tools, however, often contain a significant amount of errors caused by the use of general heuristics that are bound to fail in certain situations. Due to this fact, a manual revision of the mappings created by matching systems is often inevitable to guarantee the quality of the integration.

Revising mappings is a very complex and difficult problem even for experts in the area. We can identify two sources of complexity that make mapping revision hard for humans:

- The correctness of mappings depends on the semantics of the ontologies. This means that in principle, mapping revision forces us to completely consider the ontologies linked by the mapping. This requires some form of logical reasoning which is almost impossible to do manually due to their size and complexity.
- Individual decisions about the correctness of a suggested semantic relation can have an influence on past and future decisions making the revision of a mapping a non-monotonic process. Consistently revising a mapping therefore requires to keep track of the different dependencies which is also infeasible without adequate support.

We will illustrate these two sources of complexity using a small example. Imagine two ontologies describing scientific publications and the following semantic relations between concepts of the two ontologies:

$$\textit{1:Abstract} \quad \text{equivalent to} \quad \textit{2:Abstract} \tag{1}$$

$$\textit{1:Document} \quad \text{equivalent to} \quad \textit{2:Document} \tag{2}$$

$$\textit{1:Document} \quad \text{broader than} \quad \textit{2:Review} \tag{3}$$

At a first glance all of these relations look correct. There are situations, however, where seemingly correct relations like the first equivalence are incorrect. Taking the whole ontologies into account it turns out that the intended meaning of concept *2:Abstract* is not the one of a summary of a document as in the first ontology, but that of an abstract entity (e.g., a topic of a document). In particular, the ontologies contain the following axioms:

$$\textit{1:Abstract} \sqsubseteq \textit{1:Document} \tag{4}$$

$$\textit{2:Abstract} \sqsubseteq \neg\textit{2:Document} \tag{5}$$

$$\textit{2:Review} \sqsubseteq \textit{2:Document} \tag{6}$$

Taking into account these definitions the mismatch can be detected using the following chain of reasoning: The first equivalence relation implies that every instance of concept *2:Abstract* will also be an instance of concept *1:Document* because *1:Abstract* is its subclass. On the other hand the second equivalence relation implies that every instance of concept *1:Document* will also be an instance of concept *2:Document*. We can conclude that this will make every instance of *2:Abstract* also an instance of *2:Document*. But since the concepts *Abstract* and *Document* are defined to be disjoint in the second ontology, concept *Abstract* becomes unsatisfiable and the second ontology will therefore be incoherent.

The example also shows the dependencies between decisions. The chain of reasoning described above makes clear that the first two equivalences in our mapping cannot both be true at the same time. This means that if we first decide that the first equivalence is correct and then move on to the second equivalence and also decide that this second equivalence relation is correct, we have to revise our decision on the first one in order to avoid the model becoming inconsistent. Further, if we decide that the second equivalence is correct, then the third relation also has to be correct, because it follows from the fact that *Review* is defined as a subclass of *Document* in the second ontology.

The goal of the work reported in this paper is to develop formal methods to support the process of manual mapping revision. In particular, we address the following aspects of mechanizing this support.

- We describe a formalization of automatically generated mappings in terms of distributed description logics as a basis for reasoning about the implications of mappings.
- We propose methods for computing the implications of a revision choice and for detecting conflicts between different choices based on this formalization.
- We develop a method for making suggestions concerning the correctness of semantic relations not explicitly evaluated by the user on the basis of previous decisions and confidence estimations provided by the matching system.

In the context of this work, we focus on the dynamics of the revision process and the use of logical reasoning for dealing with these dynamics. We built on top of previous work on repairing automatically generated mappings [12] and extend this work in several directions.

1. We provide a formal model of the process of mapping revision as a basis for identifying the role of automated reasoning in the process.
2. We provide a detailed analysis of reasoning about mappings in the framework of distributed description logics as a basis for detecting inconsistency and implied mappings.
3. We present a revision method for mappings based on a modification of the diagnostic reasoning method used in [12] better suited for large mapping sets.

4. We show that the use of formal reasoning as a basis for mapping revision significantly reduces the effort involved in manual evaluation of automatically generated mappings.

The paper is organized as follows. In section 2 we first take a broader look at the problem of mapping revision. We introduce some basic terminology and formalize the process of mapping evaluation as a sequence of evaluation decisions. In section 3 we introduce distributed description logics as a formal model for representing ontologies and mappings, and show how to extend this approach towards non-standard methods for reasoning about mappings. The actual revision algorithm, which is based on the principles of diagnostic reasoning, is described in section 4. In section 5 we present the results of an experimental evaluation with respect to two different scenarios. We conclude by positioning our work in the context of related approaches and discuss the results achieved so far.

2 A Formal Model of Mapping Revision

In this section we present a formal framework for modeling mapping revision in the context of manual evaluation. The key element of the framework is the notion of an evaluation function. An evaluation function describes a partial or complete evaluation of a mapping conducted by a domain expert. Based on this conceptualization we define the notion of a valid revision function that plays a crucial role with respect to the task of automated and semi-automated mapping revision. Before turning to the description of the framework, let us preliminarily recall the notion of a mapping.

2.1 Mapping Preliminaries

A mapping between ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ can be defined as a set of correspondences. Each correspondence expresses a semantic relation between a terminological entity of $\mathcal{O}_1$ and $\mathcal{O}_2$. As described by Euzenat and Shvaiko in [5], a correspondence can be defined as follows.

Definition 1 (Correspondence) *Given ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, let Q be a function that defines sets of matchable elements $Q(\mathcal{O}_1)$ and $Q(\mathcal{O}_2)$ of ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively. Then a* correspondence *is a 4-tuple $\langle e, e', r, n\rangle$ such that $e \in Q(\mathcal{O}_1)$ and $e' \in Q(\mathcal{O}_2)$, r is a semantic relation, and n is a confidence value from a suitable structure $\langle D, \leqslant\rangle$.*

The generic form of definition 1 allows to capture a wide class of correspondences by varying what is admissible as matchable element, semantic relation, and confidence value. In this work, we impose the following additional restrictions on correspondences: We only consider correspondences between concepts. We also restrict r to be one of the semantic relations from the set $\{\equiv, \sqsubseteq, \sqsupseteq\}$. In other words, we only focus on equivalence and subsumption correspondences. Given concepts $A \in Q(\mathcal{O}_1)$ and $B \in Q(\mathcal{O}_2)$ subsumption correspondence $\langle A, B, \sqsubseteq, 1.0\rangle$ is correct if everything that we account to be an instance of *A* also has to be accounted to be an instance of *B*. The

equivalence relation is defined as subsumption in both directions. Finally, we assume that the confidence value is represented numerically on $D = [0.0, 1.0]$.

Notice that the confidence value n can be seen as a measure of trust in the fact that the correspondence holds. The higher the confidence degree with regard to the ordering $\leqslant$ the more likely relation r holds between matchable elements e and e'. Given a set of semantic correspondences, we can define the notion of a mapping as a container of these semantic correspondences.

Definition 2 (Mapping) *Given ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, let Q be a function that defines sets of matchable elements $Q(\mathcal{O}_1)$ and $Q(\mathcal{O}_2)$ of ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively. $\mathcal{M}$ is a mapping between $\mathcal{O}_1$ and $\mathcal{O}_2$ iff for all correspondences $\langle e, e', r, n\rangle \in \mathcal{M}$ we have $e \in Q(\mathcal{O}_1)$ and $e' \in Q(\mathcal{O}_2)$.*

Based on this general model of a mapping, we can now define the process of mapping revision.

2.2 Revising Mappings

Let us consider an integration scenario of two overlapping ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. In order to perform the integration a mapping $\mathcal{M}$ has to be generated and evaluated by a domain expert. For each correspondence in $\mathcal{M}$ the evaluator has to choose between three alternatives — $correct$, $incorrect$ and $unknown$. By default, each correspondence is implicitly evaluated as $unknown$ as long as no evaluation is available for it. Having this setting, the evaluation process can be modeled as a function e that assigns to each correspondence of a given mapping a value from the set $\{correct, incorrect, unknown\}$.

Definition 3 (Evaluation function) *An evaluation function $e : \mathcal{M} \rightarrow \{correct, incorrect, unknown\}$ is defined by*

$$e(c) \mapsto \begin{cases} correct & \text{if } c \text{ is accepted} \\ incorrect & \text{if } c \text{ is rejected} \\ unknown & \text{otherwise} \end{cases} \qquad \text{for all } c \in \mathcal{M}$$

Furthermore, let $e(\mathcal{M}, v) \subseteq \mathcal{M}$ be defined as $e(\mathcal{M}, v) = \{c \in \mathcal{M} | e(c) = v\}$ for all $v \in \{correct, incorrect, unknown\}$.

In a typical scenario mapping evolution is a sequential process that starts from $e(\mathcal{M}, unknown) = \mathcal{M}$. By iteratively evaluating correspondences in the mapping finally a complete evaluation with $e(\mathcal{M}, unknown) = \emptyset$ is achieved. In order to model such a stepwise evolution we further introduce a notion of a successor of an evaluation function e, which is defined as an evaluation function containing more expert evaluations compared to e.

Definition 4 (Successor evaluation function) *Given an evaluation function e, an evaluation function e' is called a successor of e when $e(\mathcal{M}, correct) \subseteq e'(\mathcal{M}, correct)$, $e(\mathcal{M}, incorrect) \subseteq e'(\mathcal{M}, incorrect)$ and $e(\mathcal{M}, unknown) \supset e'(\mathcal{M}, unknown)$.*

We refer to the process of choosing a subset $\mathcal{M}'$ of $\mathcal{M}$ based on a partial evaluation as mapping revision and model this process as applying a revision function rev. The input arguments of rev are a mapping $\mathcal{M}$, the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ and an evaluation function e defined for $\mathcal{M}$. By applying rev the input mapping $\mathcal{M}$ will be divided into a mapping $\mathcal{M}' \subseteq \mathcal{M}$ of selected correspondences and a mapping $\mathcal{M} \setminus \mathcal{M}'$ of unselected correspondences. Furthermore, an (possibly) extended evaluation function e' will be computed.

Definition 5 (Revision function) *Given an evaluation function e, a revision function rev is defined as a function $rev(\mathcal{M}, \mathcal{O}_1, \mathcal{O}_2, e) = \langle \mathcal{M}', e' \rangle$ such that $\mathcal{M}' \subseteq \mathcal{M}$ and $e' = e$ or e' is a successor of e. Further we say that rev selects a correspondence c iff $c \in \mathcal{M}'$ and unselects c iff $c \in \mathcal{M} \setminus \mathcal{M}'$.*

By definition 5 we merely specified the domain and co-domain of a revision function. There are at least three additional requirements that a revision function should fulfill. On the one hand a revision function should agree with the evaluation function. Every correspondence evaluated as $correct$ should be selected and every correspondence evaluated as $incorrect$ should be unselected. Further, we require that the application of a revision function should result in a consistent mapping. Finally, we want all correspondences that logically follow from the selected correspondences to be selected, too.

Definition 6 (Valid revision function) *Given an evaluation function e, a revision function $rev(\mathcal{M}, \mathcal{O}_1, \mathcal{O}_2, e) = \langle \mathcal{M}', e' \rangle$ is valid iff*

- *$e'(\mathcal{M}, correct)$ is closed under deduction,*
- *$\mathcal{M}'$ is consistent, and*
- *for all correspondences $c \in \mathcal{M}$ we have $e'(c) = correct \rightarrow c \in \mathcal{M}'$ and $e'(c) = incorrect \rightarrow c \notin \mathcal{M}'$.*

Notice that we did not specifify which correspondences in $e'(\mathcal{M}, unknown)$ should be part of $\mathcal{M}'$. Given that $e'(\mathcal{M}, unknown)$ is an inconsistent subset of $\mathcal{M}$, some elements of $e'(\mathcal{M}, unknown)$ have to be unselected to ensure the validity of the revision function. The following are two desired characteristics for guiding this choice.

Minimal Change: The additional knowledge given by an expert evaluation should be used in way to force a minimal set of modifications upon the knowledge encoded in the correspondences of the input mapping. In our context modifications amount to unselecting correspondences.

Maximal Confidence: The chosen selection should be justified by the confidence values of the correspondences of the input mapping. In particular, the set of selected correspondences $\mathcal{M}'$ has to be maximized with respect to the sum of confidence values of the correspondences in $\mathcal{M}'$.

First, notice that the criteria of *Minimal Change* and *Maximal Confidence* may result in conflicts. There might be two consistent subsets $\mathcal{M}'$ and $\mathcal{M}''$ of the input mapping

selected by a valid revision function, where $|\mathcal{M}'| > |\mathcal{M}''|$ but $\mathcal{M}''$ is maximal with respect to the sum of confidences. Secondly, for both the *Minimal Change* and the *Maximal Confidence* criterion we have to expect problems of scalability. For example, finding a consistent mapping selection with maximal confidence is a hard optimization problem. Thirdly, even though both criteria make sense from a theoretical point of view, it is not obvious that the specific problem of mapping revision will be solved in an appropriate way, if we stick too strictly to one of these criteria. Therefore, we have to construct a revision algorithm that uses the introduced criteria as guiding principles and finds a reasonable weighting between their importance and the specific characteristics of the mapping revision problem.

3 A Logical Foundation for Mapping Revision

Due to the common acceptance of OWL and DL, there exists a formal agreement on constructs for representing ontologies on the web. Contrary to this, a consensus on the theoretical formalization of mappings has not yet been reached. A straightforward way to formalize mappings would be to directly apply language constructs available in OWL and hence to interpret correspondences as DL axioms. However, this approach intermingles axioms encoding local ontological knowledge with axioms encoding mappings, making it hard to distinguish between the impact of mappings and the impact of local knowledge. Obviously, this distinction is crucial to the process of mapping revision.

In this work we propose the use of distributed description logics (DDL) as a theoretical underpinning for mapping formalization. The main conceptual benefit of DDL compared to the encoding of mappings as OWL axioms is DDL's treatment of mappings as first class entities, called bridge rules. Additional advantage of using DDL comes from its ability to accommodate reasoning with and about mappings in a truly distributed manner. Thus, it is possible to revise mappings without the necessity of integrating mappings with ontologies. Notice that this is a major benefit of DDL, because in many scenarios access to an ontology as a whole might be granted due to privacy restrictions, while reasoning services for the relevant parts of the ontology might be available. A more detailed argumentation for the use of DDL bridge rules can be found in [2].

3.1 Formalizing Mappings

Distributed description logics, as described in [22], can be understood as a formalism for representation and analysis of multiple ontologies pairwise linked by directional semantic mappings. In this context, depicted in Figure 1, a pair composed of a set $\mathbb{O}$ of ontologies and a set $\mathbb{M}$ of associated mappings between arbitrary pairs of ontologies constitutes a *distributed ontology* $\mathfrak{O} = \langle \mathbb{O}, \mathbb{M} \rangle$. In this section we recall the basics of distributed description logics shedding the light on the supported syntactical constructs for expressing ontological mappings and their respective semantic definitions.

Syntax: Let I be a set of indices. Given a collection of ontologies $\mathbb{O} = \{\mathcal{O}_i\}_{i \in I}$, the distributed description logics starts from a set $\{DL_i\}_{i \in I}$ of description logic theories.

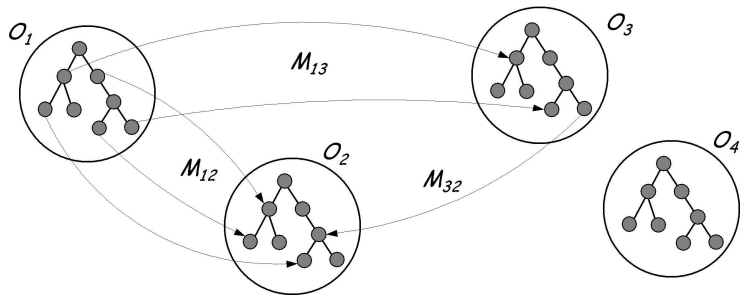

Figure 1: An example of a distributed ontology

Each ontology $\mathcal{O}_i$ is standardly formalized as a T-box $\mathcal{T}_i$ of a description logic DL_i. Therefore, it contains definitions of concepts and properties, as well as general axioms relating them to each other.[1] For distinction, every definition is prefixed with an index of T-box it belongs to, i.e., $i{:}\,C$, $i{:}\,C \sqsubseteq D$. A collection of T-boxes $\{\mathcal{T}_i\}_{i\in I}$ formally represents the collection of ontologies $\mathbb{O}$.

A collection $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i\neq j\in I}$ refers to the mappings of a distributed ontology $\mathfrak{O}$. Every mapping $\mathcal{M}_{ij}$ containing semantic correspondences between ontology $\mathcal{O}_i$ and $\mathcal{O}_j$ is respectively formalized as a set $\mathfrak{B}_{ij}$ of bridge rules between corresponding T-boxes $\mathcal{T}_i$ and $\mathcal{T}_j$. Consequently, the collection of mappings $\mathbb{M}$ is formally represented in distributed description logics by a collection of bridge rules $\{\mathfrak{B}_{ij}\}_{i\neq j\in I}$. Each bridge rule in $\mathfrak{B}_{ij}$ has a certain type and connects a concept from $\mathcal{T}_i$ to a concept of $\mathcal{T}_j$. Distributed description logics supports two core bridge rule types:

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

The derived equivalence bridge rule $i{:}\,C \xrightarrow{\equiv} j{:}\,D$ is defined as the conjunction of corresponding into and onto rules.

Intuitively, bridge rules from $\mathcal{T}_i$ to $\mathcal{T}_j$ express a subjective possibility of $\mathcal{T}_j$ to translate some of the concepts of $\mathcal{T}_i$ into its local concepts. For example, the into bridge rule $i{:}\,\mathit{PhDThesis} \xrightarrow{\sqsubseteq} j{:}\,\mathit{Thesis}$ states that concept *PhDThesis*, from the $\mathcal{T}_j$'s point of view, is less general than its local concept *Thesis*. Similarly, the onto bridge rule $i{:}\,\mathit{InProceedings} \xrightarrow{\sqsupseteq} j{:}\,\mathit{ConferencePaper}$ expresses the more generality relation.

Bridge rules are directional, hence $\mathfrak{B}_{ij}$ is not the inverse of $\mathfrak{B}_{ji}$. A certain mapping $\mathfrak{B}_{ij}$ might be also empty, which represents the impossibility of $\mathcal{T}_j$ to interpret concepts of $\mathcal{T}_i$ into any of its local concepts.

The collection of T-boxes and bridge rules between them forms a distributed T-box $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I}\rangle$ of distributed description logics, which is in turn the formalization of the whole distributed ontology $\mathfrak{O} = \langle\mathbb{O}, \mathbb{M}\rangle$.

Semantics: The semantics of distributed description logics is based on a fundamental idea that each ontology $\mathcal{T}_i$ is locally interpreted on a local interpretation domain. Each

[1] We assume the reader is familiar with description logics; an introduction can be found in [1].

local interpretation $\mathcal{I}_i$ consists of a local domain $\Delta^{\mathcal{I}_i}$ and a valuation function $\cdot^{\mathcal{I}_i}$, which maps every concept to a subset of $\Delta^{\mathcal{I}_i}$ and every role to a subset of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$. Given that setting, the first component of semantics is a set of interpretations $\{\mathcal{I}_i\}_{i \in I}$, one for each $\mathcal{T}_i$.

To resolve heterogeneity between different interpretation domains distributed description logics introduces a second semantic component, a domain relation. A domain relation r_{ij} represents a possible way of mapping the elements of $\Delta^{\mathcal{I}_i}$ into the domain $\Delta^{\mathcal{I}_j}$, such that r_{ij} denotes $\{d' \in \Delta^{\mathcal{I}_j} \mid \langle d, d' \rangle \in r_{ij}\}$; for any subset D of $\Delta^{\mathcal{I}_i}$, $r_{ij}(D)$ denotes $\bigcup_{d \in D} r_{ij}(d)$; and for any $R \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ $r_{ij}(R)$ denotes $\bigcup_{\langle d,d' \rangle \in R} r_{ij}(d) \times r_{ij}(d')$.

A distributed interpretation $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I} \rangle$ satisfies a distributed T-box $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle$, is called a model of $\mathfrak{T}$, if all its' components (T-boxes and bridge rules) are satisfied according to the following rules:

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

The notion of logical entailment in distributed description logics is defined as usual in classical description logics. $\mathfrak{T} \models i{:}\,C \sqsubseteq D$ if for every distributed interpretation $\mathfrak{I}$, $\mathfrak{I} \models \mathfrak{T}$ implies $\mathfrak{I} \models i{:}\,C \sqsubseteq D$. Given a distributed T-box $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle$, one can perform some basic distributed inferences such as checking concept satisfiability and subsumption. A concept $i{:}\,C$ is *satisfiable* with respect to $\mathfrak{T}$ if there exist a distributed model $\mathfrak{I}$ of $\mathfrak{T}$ such that $C^{\mathcal{I}_i} \neq \emptyset$. A concept $i{:}\,C$ is *subsumed* by a concept $i{:}\,D$ with respect to $\mathfrak{T}$ ($\mathfrak{T} \models i{:}\,C \sqsubseteq D$) if for every distributed model $\mathfrak{I}$ of $\mathfrak{T}$ we have that $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$.

Encoding correspondences as bridge rules: To formalize semantic mappings in terms of presented distributed description logics we follow the approach of encoding correspondences as bridge rules. In particular, each correspondence $\langle e, e', r, n \rangle$ between a pair of ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ is translated into a set of bridge rules using a translation function t in the following way:

$$t(\langle id, e, e', \sqsubseteq, n \rangle) = i{:}\,e \xrightarrow{\sqsubseteq} j{:}\,e', n \quad \wedge \quad j{:}\,e' \xrightarrow{\sqsupseteq} i{:}\,e, n$$
$$t(\langle id, e, e', \sqsupseteq, n \rangle) = i{:}\,e \xrightarrow{\sqsupseteq} j{:}\,e', n \quad \wedge \quad j{:}\,e' \xrightarrow{\sqsubseteq} i{:}\,e, n$$

Equivalence correspondences are interpreted as a pair of inclusion correspondences which are treated individually in our experiments.

3.2 Reasoning with mappings

Reasoning in distributed description logics is founded on exploitation of the capability of bridge rules to propagate knowledge across interlinked ontologies. Such a propagation from an ontology $\mathcal{T}_i$ (the source) to ontology $\mathcal{T}_j$ (the target) via a set of bridge

rules $\mathfrak{B}_{ij}$ from i to j can be represented by a pattern of the following form:

$$\frac{(1)\text{ axioms in } i \quad \text{and} \quad (2)\text{ bridge rules from } i \text{ to } j}{(3)\text{ axiom in } j}$$

which must be read as: if axioms (1) are satisfied in $\mathcal{T}_i$ and the bridge rules (2) are contained in $\mathfrak{B}_{ij}$, then the axiom (3) must be satisfied in $\mathcal{T}_j$.

Now, following the semantics of bridge rules, one can observe that an interaction between onto and into bridge rules can indeed cause the effect of propagating concept subsumption axioms:

$$\frac{i{:}\,A \sqsubseteq B \quad \text{and} \quad i{:}\,A \xrightarrow{\sqsupseteq} j{:}\,G,\ i{:}\,B \xrightarrow{\sqsubseteq} j{:}\,H}{j{:}\,G \sqsubseteq H} \tag{7}$$

Because $G^{\mathcal{I}_j} \subseteq r_{ij}(A^{\mathcal{I}_i}) \subseteq r_{ij}(B^{\mathcal{I}_i}) \subseteq H^{\mathcal{I}_j}$, we indeed can derive that $j{:}\,G \sqsubseteq H$.

Formally, the subsumption propagation pattern (7) can be stated as follows: Given a distributed ontology $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I}\rangle$, if $\mathfrak{B}_{ij}$ contains a pair of bridge rules $i{:}\,A \xrightarrow{\sqsupseteq} j{:}\,G$ and $i{:}\,B \xrightarrow{\sqsubseteq} j{:}\,H$, then $\mathfrak{T} \models i{:}\,A \sqsubseteq B \Longrightarrow \mathfrak{T} \models j{:}\,G \sqsubseteq H$.

In languages that support the disjunction construct, propagation rule (7) can be generalized to the propagation of subsumption axioms between a concept and a disjunction of $n \geqslant 0$ concepts in the following way:

$$\frac{i{:}\,A \sqsubseteq B_1 \sqcup \ldots \sqcup B_n \quad \text{and} \quad i{:}\,A \xrightarrow{\sqsupseteq} j{:}\,G,\ i{:}\,B_k \xrightarrow{\sqsubseteq} j{:}\,H_k\ (1 \leqslant k \leqslant n)}{j{:}\,G \sqsubseteq H_1 \sqcup \ldots \sqcup H_n} \tag{8}$$

The important property of propagation (8) is that it is the most general form of subsumption propagation in distributed description logics. Moreover, it has been shown in [21] that in case of $\mathcal{SHIQ}$ component ontologies, by adding this propagation pattern as an additional inference rule to $\mathcal{SHIQ}$ description logics tableau one gets a correct and complete distributed tableaux method for reasoning with distributed ontologies. A detailed description of the distributed tableau algorithm can be found in [22].

3.3 Reasoning about mappings

In the previous subsection we have seen how to exploit the knowledge encoded in the mapping of a distributed reasoning scenario. Building on that, in the following we introduce several properties to describe characteristics of mappings. We argue that these properties are well suited to cope with changing and evolving mappings.

Let us reconsider the example from the introduction. Suppose there are two ontologies $\mathcal{T}_1$ and $\mathcal{T}_2$ about the domain of libraries and bridge rules mapping $\mathfrak{B}_{12}$ from $\mathcal{T}_1$ to $\mathcal{T}_2$ automatically generated by matching system. Amongst others we have bridge rules (9) and (10) as elements of this mapping.

$$1{:}\,\mathit{Abstract} \xrightarrow{\sqsupseteq} 2{:}\,\mathit{Abstract} \tag{9}$$

$$1{:}\,\mathit{Document} \xrightarrow{\sqsubseteq} 2{:}\,\mathit{Document} \tag{10}$$

Further, let $1{:}\,Abstract \sqsubseteq 1{:}\,Document$ be in $\mathcal{T}_1$ and $2{:}\,Document \sqsubseteq \neg 2{:}\,Abstract$ in $\mathcal{T}_2$. By applying (9) and (10) we can conclude that $2{:}\,Abstract \sqsubseteq 2{:}\,Document$. Thus, concept $2{:}\,Abstract$ is distributed unsatisfiable and therefore $\mathfrak{B}_{12}$ is inconsistent. The accordant property of a mapping can be defined as follows.

Definition 7 (Consistency) *Given $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I}\rangle$, $\mathfrak{B}_{ij}$ is consistent with respect to $j{:}\,C$ iff $\mathcal{T}_j \not\models C \sqsubseteq \bot \rightarrow \mathfrak{T} \not\models j{:}\,C \sqsubseteq \bot$. Otherwise $\mathfrak{B}_{ij}$ is inconsistent with respect to $j{:}\,C$. $\mathfrak{B}_{ij}$ is consistent with respect to $\mathcal{T}_j$ iff for all $j{:}\,C$ $\mathfrak{B}_{ij}$ is consistent with respect to $j{:}\,C$. Otherwise $\mathfrak{B}_{ij}$ is inconsistent with respect to $\mathcal{T}_j$.*

Algorithm 1 decides consistency of a mapping with respect to terminology $\mathcal{T}_i$ of $\mathfrak{T}$ and can be understood as a straight forward way to implement definition 7.

Algorithm 1

IsConsistent($\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I}\rangle, i$)

```
1: for all C ∈ GetAllConcepts(𝔗, i) do
2:   if 𝒯_i ⊭ C ⊑ ⊥ and 𝔗 ⊨ i:C ⊑ ⊥ then
3:     return false
4:   end if
5: end for
6: return true
```

Obviously, inconsistency is a clear symptom for defective parts in a mapping $\mathfrak{B}_{ij}$ of a distributed terminology. We can thus conclude that at least one bridge rule in $\mathfrak{B}_{ij}$ has to be incorrect, given that $\mathfrak{B}_{ij}$ is inconsistent.

Can we find an analogue principle to determine wether a bridge rules b follows from $\mathfrak{B}_{ij}$? Obviously, b follows from $\mathfrak{B}_{ij}$ if and only if b does not provide any additional pieces of information that are not explicit or implicit available in $\mathfrak{B}_{ij}$. Consider again the example from the introduction where $\mathcal{T}_2$ contains the axiom *Document* $\sqsubseteq$ *Review* and consider the following bridge rules.

$$1{:}\,\textit{Document} \xrightarrow{\sqsupseteq} 2{:}\,\textit{Document} \tag{11}$$

$$1{:}\,\textit{Document} \xrightarrow{\sqsupseteq} 2{:}\,\textit{Review} \tag{12}$$

The second bridge rule follows from the first one, because every distributed interpretation that is a model for the first bridge rule will also be a model for the second bridge rule. The following definition formally introduces the corresponding notion of entailment in general.

Definition 8 (Entailment) *Given $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I}\rangle$. A bridge rule b is entailed by $\mathfrak{T}$ iff every model $\mathfrak{I}$ of $\mathfrak{T}$ satisfies b.*

While this definition precisely captures the notion of entailment, the distributed tableaux algorithm cannot be used directly to check if a certain bridge rule is entailed from a set of bridge rules. We solve this problem by introducing the notion of a concepts image. The image $C^{i\rightarrow j}$ 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. An image $C^{i\to j}$ in $\mathcal{T}_j$ can be understood as the counterpart of concept $i{:}\,C$ in $\mathcal{T}_j$. The image of a concept can alternatively be defined by extending $\mathfrak{T}$ 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 $\mathfrak{B}_{ij}$. Since $C^{i\to j}$ is linked to $i{:}\,C$ via the equivalence bridge rule, we have $r_{ij}(C^{\mathcal{I}_i}) \equiv (C^{i\to j})^{\mathcal{I}_j}$ for any model $\mathfrak{I}$ of $\mathfrak{T}$. The following is the formal representation of this constructive definition.

Definition 9 (Image extension) *Given $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I}\rangle$. Let $\mathcal{T}'_j$ be defined as $\mathcal{T}'_j = \mathcal{T}_j \cup \{\top \sqsupseteq C^{i\to j}\}$ $\mathcal{T}_j$ and let $\mathfrak{B}'_{ij}$ be defined as $\mathfrak{B}'_{ij} = \mathfrak{B}_{ij} \cup \{i{:}\,C \xrightarrow{\equiv} j{:}\,C^{i\to j}\}$. Then $j{:}\,C^{i\to j}$ is called the image of $i{:}\,C$ and $\mathfrak{T}^{C^{i\to j}} = \langle\{\mathcal{T}_k\}_{k\neq j,k\in I} \cup \{\mathcal{T}'_j\}, \{\mathfrak{B}_{kl}\}_{k\neq i,l\neq j,k\neq l\in I} \cup \{\mathfrak{B}'_{ij}\}\rangle$ is called $\mathfrak{T}$ extended by $C^{i\to j}$.*

Remember that entailment of a bridge rule b is based on the fact that every model of $\mathfrak{T}$ satisfies b. Based on the following proposition we can in a straight forward way define a procedure for deciding entailment of a bridge rule. The correctness of this proposition follows directly from the definition of image extension.

Proposition 1 (Bridge rule equivalence) *Given $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I}\rangle$, for each pair of concepts $\langle i{:}\,C, j{:}\,D\rangle$ with $i \neq j \in I$ we have.*

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

Algorithm 2 makes use of proposition 1. It decides entailment by reasoning in the extended terminology $\mathfrak{T}^{C^{i\to j}}$.

Algorithm 2

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

```
1: if R = ⊑ then
2:     return 𝔗^{C^{i→j}} ⊨ j:C^{i→j} ⊑ D
3: else if R = ⊒ then
4:     return 𝔗^{C^{i→j}} ⊨ j:C^{i→j} ⊒ D
5: else if R = ≡ then
6:     return 𝔗^{C^{i→j}} ⊨ j:C^{i→j} ≡ D
7: end if
```

4 Implementing a Revision Function

Remember that a revision function has to provide two components. On the one hand we would like to extend the evaluation as much as possible. On the other hand we

have to rationally select a subset of the input mapping taking into account both the evaluation function as well as the confidence values of the correspondences evaluated as $unknown$. We start with the extension of the evaluation function in section 4.1 and continue with the selection component in the following subsections.

Up to now, we have introduced the evaluation framework based on our intuitive understanding of a correspondence. Due to the formal encoding of correspondences in bridge rules of distributed description logics , an evaluation function can now be seen as a function that assigns the values $\{correct, incorrect, unknown\}$ to bridge rules, and a revision function takes as input the T-boxes of a distributed terminology as well as a set of bridge rules between them. In the same way we will also transfer the other definitions introduced in section 2.2 to the context of distributed description logics.

4.1 Extension

The extension of the evaluation function can be implemented in a straight forward way by applying the algorithms for checking consistency and entailment (algorithm 1 and 2). Remember that, given an evaluation function e for a set of bridge rules $\mathfrak{B}_{ij}$ between $\mathcal{T}_i$ and $\mathcal{T}_j$, $\mathfrak{B}_{ij}$ is divided in three complementary subsets $e(\mathfrak{B}_{ij}, correct)$, $e(\mathfrak{B}_{ij}, incorrect)$ and $e(\mathfrak{B}_{ij}, unknown)$. Since all bridge rules in $e(\mathfrak{B}_{ij}, correct)$ are accepted, we can use this information to derive that certain bridge rules in $e(\mathfrak{B}_{ij}, unknown)$ have also implicitly been evaluated, even though the evaluator might not be aware of this. On the one hand, for each bridge rule b with $e(b) = unknown$ we know that b has to be evaluated as $correct$ if b can be entailed by $e(\mathfrak{B}_{ij}, correct)$. On the other hand, we can conclude that each bridge rule b with $e(b) = unknown$ has to be evaluated as $incorrect$ if $e(\mathfrak{B}_{ij}, correct) \cup b$ is inconsistent.

Algorithm 3

EXTENDEVALUATION($\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I} \rangle, e, k, l$)

1: $e' \leftarrow e$
2: $\{\mathfrak{B}_{\neg kl}\} \leftarrow \{\mathfrak{B}_{ij}\}_{i\neq j\in I} \setminus \{\mathfrak{B}_{kl}\}$
3: **for all** $b \in e(\mathfrak{B}_{kl}, unknown)$ **do**
4: **if** ISENTAILED($\langle \{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{\neg kl}\} \cup \{e(\mathfrak{B}_{kl}, correct)\} \rangle, b$) **then**
5: $e'(b) \leftarrow correct$
6: **end if**
7: **if** ISCONSISTENT($\langle \{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{\neg kl}\} \cup \{e(\mathfrak{B}_{kl}, correct) \cup \{b\}\} \rangle, l$) **then**
8: $e'(b) \leftarrow incorrect$
9: **end if**
10: **end for**
11: **return** e'

Algorithm 3 is a direct implementation of this strategy. This algorithm takes as input a distributed terminology $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i\in I}, \{\mathfrak{B}_{ij}\}_{i\neq j\in I} \rangle$, an evaluation function e defined for mapping $\mathfrak{B}_{kl}$, and the indices $k, l \in I$ referring to terminologies $\mathcal{T}_k$ and $\mathcal{T}_l$ respectively. Though this approach requires reasoning in a modified distributed terminology, all modifications are related to the mapping attached to $\mathcal{T}_l$. This means

that the algorithm can be executed locally on the reasoner hosting $\mathcal{T}_l$. The same holds for the algorithms introduced in the following subsections. Notice, that this is an important aspect, because in a realistic scenario mappings will be managed locally and modifications of mappings and terminologies hosted by different reasoning peers will not be granted, in general. While we primarily discussed the case where we have two ontologies connected via a mapping, algorithm 3 makes use of the information encoded in the whole distributed terminology $\mathfrak{T}$ which might consist of several ontologies and mappings between them. Thus, it might happen that the revision process is not only directly affected by the axioms in $\mathcal{T}_k$ and $\mathcal{T}_l$ but also indirectly by some of the other ontologies and mappings.

Let us revisit the small example introduced in an informal way in section 1 to better understand the capabilities of extending an evaluation function. This example will illustrate two essential issues about extending an evaluation.

Example 1 *Given the bridge rule mapping $\mathfrak{B}_{12}$ from $\mathcal{T}_1$ to $\mathcal{T}_2$ consisting, amongst others, of the following bridge rules generated by a fully automatized matching system.*

$$1\text{:}\,\textit{Document} \xrightarrow{\equiv} 2\text{:}\,\textit{Document},\ 0.98 \tag{13}$$

$$1\text{:}\,\textit{Abstract} \xrightarrow{\equiv} 2\text{:}\,\textit{Abstract},\ 0.93 \tag{14}$$

$$1\text{:}\,\textit{Document} \xrightarrow{\sqsupseteq} 2\text{:}\,\textit{Review},\ 0.57 \tag{15}$$

Suppose now that a domain expert for knowledge management evaluates $\mathfrak{B}_{12}$ starting with bridge rule $b_{(13)}$. He accepts this correspondence and thus we have $e(\mathfrak{B}_{12}, correct) = \{b_{(13)}\}$ and $e(\mathfrak{B}_{12}, unknown) = \{b_{(14)}, b_{(15)}\}$. Given the following axioms for $\mathcal{T}_1$ and $\mathcal{T}_2$

$$\begin{aligned} \mathcal{T}_1 &\models \textit{Document} \sqsupseteq \textit{Abstract} \\ \mathcal{T}_2 &\models \textit{Document} \sqsubseteq \neg\textit{Abstract} \\ \mathcal{T}_2 &\models \textit{Document} \sqsupseteq \textit{Review} \end{aligned}$$

applying the extension algorithm will result in the extended evaluation function e'_p with

$$\begin{aligned} e'_p(\mathfrak{B}_{12}, correct) &= \{b_{(13)}, b_{(15)}\} \\ e'_p(\mathfrak{B}_{12}, incorrect) &= \{b_{(14)}\} \\ e'_p(\mathfrak{B}_{12}, unknown) &= \emptyset \end{aligned}$$

Thus, for our example, we ended up with a fully evaluated mapping by applying the extension algorithm.

This example sheds light on two important aspects. On the one hand it might happen that the extension of an evaluation function results in a relatively high number of evaluation decisions that can be skipped. In this example for one evaluation decision we gained two further decisions without (direct) manual intervention. On the other hand applying the extension algorithm might sometimes result in non trivial extensions, in particular where manual evaluation might result in erroneous decisions. The incorrectness of bridge rule (14) can be counted as an example. By merely looking at the concept names, not taking their conceptual context into account, an inattentive evaluator might make a mistake that can be avoided logical reasoning .

4.2 Selection

We will now turn our attention to the selection component of the revision function that decides which parts of the input mappings evaluated as $unknown$ should be selected and unselected, respectively. First, we concentrate on the case where no expert evaluation is available. We will later see, that our approach can be extended in a straight forward way to the general case where partial evaluation is available.

In the following we rely on the classical definition of diagnosis introduced by Reiter [17]. The basic assumption of our approach is that bridge rules model semantic correspondences between concepts of different ontologies without introducing inconsistencies. A diagnosis task is normally defined in terms of a set of components $COMP$ in which a fault might have occurred, a system description SD defining the behavior of the system and a set of observations OBS (or symptoms). A diagnosis is now defined as the minimal set $\Delta \subseteq COMP$ such that the observations OBS are explained by a subset of the components having abnormal behavior. In our context we regard a bridge rule mapping $\mathfrak{B}_{ij}$ to be the set of components to be diagnosed, while $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle$ provides the system description. Observations are provided in terms of implied subsumption relations between concepts in the two ontologies. Bridge rules are assumed to be abnormal if they cause inconsistency. In other words, a diagnosis is the minimal set of bridge rules $\Delta \subseteq \mathfrak{B}_{ij}$ such that the mapping $\mathfrak{B}_{ij} \setminus \Delta$ is consistent.

Definition 10 (Diagnosis) *Given $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle$, a diagnosis for a mapping $\mathfrak{B}_{ij}$ is defined as the minimal set $\Delta \subseteq \mathfrak{B}_{ij}$ such that $\{\mathfrak{B}_{ij}\}_{i \neq j \in I} \setminus \Delta$ is consistent with respect to $\mathcal{T}_j$.*

Since we want our revision function to chose a consistent subset of $\mathfrak{B}_{ij}$ that still contains as much bridge rules as possible, we thus have to find a diagnosis. Nevertheless, computing diagnoses (minimal sets of abnormal components) is known to be computational intractable in the general case as the set of all possible diagnoses form a combinatorial search space which is exponential in the size of $COMP$. In order to deal with this problem, we adopt the notion of conflict sets [17] for guiding the search for abnormal correspondences.

Reiter defines a conflict set as a subset of the system components that together produce an abnormal behavior. In our context a conflict set is a subset of the mapping that is inconsistent. This definition implies that any inconsistent mapping automatically becomes a conflict set. This trivial conflict set, however, does not provide us with any hints about the set of bridge rules that constitute the diagnosis. In diagnosis we are normally interested in minimal conflict sets (conflict sets with the additional property that none of its subsets is a conflict set). These sets have the beneficial property that the problem caused by a minimal conflict set can be repaired by removing one component in the set. Having identified a minimal conflict set, the decision which bridge rule to remove can in a straight forward way be based on the order of confidence values. We already explained that a confidence value can be seen as a measure of trust in the fact that the correspondence holds. If we now know, given a set of bridge rules $\mathfrak{B}$, that at least one of these bridge rules has to be incorrect, the most reasonable decision is to remove the one with the lowest confidence.

A classical algorithm for computing conflict sets to find a diagnosis is based on the notion of a conflict set tree [4]. Conflict set trees allow non-redundant searching in the power set of a given axiom set, in our context the power set of a set of bridge rules. Having once identified minimal conflict sets, it is possible to compute a hitting set of these conflict sets. Notice that approaches based on the computation of a hitting set are not efficient enough for our problem. We have also tried this approach and the results of our experiments indicate that it does not scale well. Thus, we have to make use of the following observation. Given a mapping $\mathfrak{B}_{kl}$ with $k, l \in I$ and a bridge rule $b \notin \mathfrak{B}_{kl}$ from $\mathcal{T}_k$ to $\mathcal{T}_l$. Suppose that $\mathfrak{B}_{kl}$ is consistent with respect to $\mathcal{T}_l$, while $\mathfrak{B}_{kl} \cup b$ is inconsistent with respect to $\mathcal{T}_l$. We can conclude that (1) there exists at least one minimal non empty conflict set $\mathfrak{C} \subset \mathfrak{B}_{kl} \cup b$ and that (2) for every minimal conflict set $\mathfrak{C} \subseteq \mathfrak{B}_{kl}$ we have $b \in \mathfrak{C}$. This observation can be used for an algorithm that computes a diagnoses Δ based on the notion of minimal conflict sets without explicitly computing them.

First we sort all bridge rules in $\mathfrak{B}_{kl}$ descending with respect to their confidence values. Then we start with an empty set $\mathfrak{B}^*_{kl}$ adding step by step bridge rules $b \in \mathfrak{B}_{kl}$. In each step we have to check if the distributed terminology, where we temporarily replace $\mathfrak{B}_{kl}$ with $\mathfrak{B}^*_{kl}$, has become inconsistent. In case of inconsistency, we know that the current bridge rule b is an element of at least one minimal conflict set $\mathfrak{C} \subseteq \mathfrak{B}^*_{kl}$ and that there exists no bridge rule $b' \in \mathfrak{C}$ such that the confidence of b' is less than the confidence of b. We can conclude that there exists a diagnosis Δ with $b \in \Delta$. We remove b from $\mathfrak{B}^*_{kl}$ and continue with the next bridge rule. In case of no inconsistency we continue with the next step of the iteration without removing b from $\mathfrak{B}^*_{kl}$. Finally, we end up with a consistent set of bridge rules $\mathfrak{B}^*_{kl}$ and know that $\Delta = \mathfrak{B}_{kl} - \mathfrak{B}^*_{kl}$ is a diagnosis for the given problem.

Now suppose that a partial evaluation e for a mapping $\mathfrak{B}_{kl}$ is available with $e(\mathfrak{B}_{kl}, \mathit{correct}) \neq \emptyset$. The previously described algorithm can be extended in a natural way to cope with this situation. Instead of starting with an empty mapping $\mathfrak{B}^*_{kl} = \emptyset$, we have to start with $\mathfrak{B}^*_{kl} = e(\mathfrak{B}_{kl}, \mathit{correct})$. Algorithm 4 is an implementation of this approach that computes a diagnosis for the mapping $\mathfrak{B}_{kl}$ from $\mathcal{T}_k$ to $\mathcal{T}_l$ with $k \neq l \in I$. Notice, that parts of the diagnosis have already been conducted by the manual evaluation e. These are the bridge rules $e(\mathfrak{B}_{kl}, \mathit{incorrect})$. Thus, algorithm 4 returns also $e(\mathfrak{B}_{kl}, \mathit{incorrect})$ as part of the diagnosis Δ.

4.3 Revision

We can now combine algorithm 3 and 4 into a revision function (see algorithm 5). First of all, before computing the evaluation extension and the diagnosis, consistency of the evaluation function e has to be checked. Consistency of the manual evaluation is a necessary precondition for further computation. If this precondition is satisfied, the algorithm first extends the manual evaluation and then computes a diagnosis based on the extended evaluation. It returns a pair that consists of the extended evaluation and the set of selected bridge rules.

In case of an inconsistent human evaluation algorithm 5 throws an exception. This exception can be handled in the following way. Inform the evaluator about the problem and apply a modified version of algorithm 4 on the set of accepted bridge rules

Algorithm 4

COMPUTEDIAGNOSIS($\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle, e, k, l$)

1: $\{\mathfrak{B}_{\neg kl}\} \leftarrow \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \setminus \{\mathfrak{B}_{kl}\}$
2: $\mathfrak{B}^*_{kl} \leftarrow e(\mathfrak{B}_{kl}, correct)$
3: $\mathfrak{B}'_{kl} \leftarrow e(\mathfrak{B}_{kl}, unknown)$
4: SORTDESCENDING($\mathfrak{B}'_{kl}$)
5: **for all** $b \in \mathfrak{B}'_{kl}$ **do**
6: $\quad \mathfrak{B}^*_{kl} \leftarrow \mathfrak{B}^*_{kl} \cup \{b\}$
7: $\quad$ **if not** ISCONSISTENT($\langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{\neg kl}\} \cup \{\mathfrak{B}^*_{kl}\} \rangle, l$) **then**
8: $\quad\quad \mathfrak{B}^*_{kl} \leftarrow \mathfrak{B}^*_{kl} \setminus \{b\}$
9: $\quad$ **end if**
10: **end for**
11: **return** $(\mathfrak{B}'_{kl} \setminus \mathfrak{B}^*_{kl}) \cup e(\mathfrak{B}_{kl}, incorrect)$

$e(\mathfrak{B}_{kl}, correct)$ where all of these bridge rules are now treated as $unknown$. The resulting diagnosis is then presented to the evaluator in order to reject one of the bridge rules that previously have been accepted. This prodedure has to be repeated until every inconsistency in $e(\mathfrak{B}_{kl}, correct)$ is removed.

Algorithm 5

REVISE($\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle, e, k, l$)

1: $\{\mathfrak{B}_{\neg kl}\} \leftarrow \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \setminus \{\mathfrak{B}_{kl}\}$
2: **if not** ISCONSISTENT($\langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{\neg kl}\} \cup \{e(\mathfrak{B}_{kl}, correct)\} \rangle, l$) **then**
3: $\quad$ **throw** INCONSISTENCYEXCEPTION
4: **end if**
5: $e' \leftarrow$ EXTENDEVALUATION($\mathfrak{T}, e, k, l$)
6: $\mathfrak{B}'_{kl} \leftarrow$ COMPUTEDIAGNOSIS($\mathfrak{T}, e', k, l$)
7: **return** $\langle e', \mathfrak{B}_{kl} \setminus \mathfrak{B}'_{kl} \rangle$

In section 2.2 we introduced two desirable characteristics of a revision function. We referred to these characteristics as *Minimal Change*, *Maximal Confidence*. We already argued that *Minimal Change* and *Maximal Confidence* disagree in many situations. On the one hand the diagnosis Δ computed with algorithm 4 is minimal in the sense that there exists no other $\Delta' \subset \Delta$ that would also be a diagnosis. On the other hand this does not imply that there exists no other diagnosis Δ^* with $|\Delta^*| < |\Delta|$. We can conclude that algorithm 4 does not fulfill the requirement of *Minimal Change*. Due to the greediness of algorithm 4 the selection made by the revision will also in general not be of *Maximal Confidence*. Nevertheless, our algorithm results in an approximation of both criteria and yields in many cases better solutions than strict *Minimal Change* or *Maximal Confidence* algorithms.

Example 2 *Given a mapping $\mathfrak{B}_{12}$ from $\mathcal{T}_1$ to $\mathcal{T}_2$ consisting of the following bridge*

rules.

$$1{:}\,\textit{Person} \xrightarrow{\equiv} 2{:}\,\textit{Person},\ 1.0 \tag{16}$$

$$1{:}\,\textit{Reviewer} \xrightarrow{\equiv} 2{:}\,\textit{Review},\ 0.65 \tag{17}$$

$$1{:}\,\textit{Participant} \xrightarrow{\equiv} 2{:}\,\textit{ParticipationFee},\ 0.6 \tag{18}$$

Further, we have the following disjointness axioms

$$\mathcal{T}_2 \models Person \sqsubseteq \neg Review \tag{19}$$

$$\mathcal{T}_2 \models Person \sqsubseteq \neg ParticipationFee \tag{20}$$

that make the subset $\{16, 17\}$ *as well as the subset* $\{16, 18\}$ *inconsistent. In this situation our algorithm will compute* $\Delta = \{17, 18\}$ *as diagnosis. Contrary to this, a solution that fulfills the Minimal Change criteria as well as the Maximal Confidence criteria would result in a diagnosis* $\Delta = \{16\}$*. In this situation, applying the criteria would result in throwing away the only correct bridge rule in the set which is clearly not what we want.*

Notice that, based on our experience, similar patterns frequently occur in many mappings. Opposed to the theoretical attractiveness, this example clearly shows that in practice the global *Minimal Change* and *Maximal Confidence* criteria fails, while the suggested greedy approach succeeds.

5 Experiments

In the experimental section we show that our approach can be successfully applied to different scenarios of mapping evolution. In section 5.1 we first focus on the case when there is no expert evaluation available. In such a situation the revision process will be fully automated. In section 5.2 we deal with the case where a domain expert is available and a complete expert evaluation has to be achieved. In such a scenario we have to focus on the extension of the evaluation function. In particular, we discuss the amount of effort that will be saved due to automatized extension of the evaluation function.

We evaluated our approach using the OntoFarm Dataset of ontologies in the domain of conference organization [25][2] which is part of the Ontology Alignment Evaluation Initiative (OAEI) the de facto standard for evaluating ontology matching approaches. We chose this particular data set because it is the largest collection of OWL ontologies about the same topic that still allow for a complete evaluation of the results. The dataset of the benchmark track which is often used in comparable studies is much smaller both in terms of the number and the size of ontologies involved. There are larger datasets but out of these only the anatomy data set provides a reference mapping. Further, all data sets except for the benchmark and the conference data set that we use are not really OWL ontologies, but only consist of simple hierarchies. This means that

[2]The ontologies are available from `http://nb.vse.cz/~svabo/oaei2006/`.

the OntoFarm dataset so far is the best and most representative benchmark for testing methods supporting the alignment of OWL ontologies.

Meanwhile the OntoFarm Dataset consists of 13 ontologies six of which we used in our experiments. These are the ontologies CMT, CRS, PCS, CONFTOOL, SIGKDD and EKAW, containing between 17 and 77 concepts. The other ontologies where omitted because not all of the matching systems in our experiments were able to provide the corresponding mappings or because the ontologies have been added to the dataset after we designed the experiment.

5.1 Revising without Expert Evaluation

In a first experiment reported in [12] we considered pairwise mappings produced between the six ontologies presented above. We manually created a reference mapping used to evaluate the automatically generated revision. In order to guarantee the fairness of evaluation, we had three people individually checking the mappings. In cases of a disagreement the correctness of a correspondence was decided by a majority vote. It turned out that there was very little disagreement with respect to the correctness of correspondences. For only about 3% of the correspondences the result had to be determined by vote.

For each pair of ontologies, we ran algorithm 4 in both directions. Table 1 summarizes the results with respect to precision of four state of the art matching systems[3] and precision and recall of the debugging algorithm. The precision of the debugging algorithm ranges between 78% and 100%. These results can be compared to the precision of the matcher and the strategy to remove correspondences randomly. Applying our recommendations to Falcon-AO and OWL-CTXmatch we were able to increase the precision by 2% and 6%, respectively. In the case of matcher Falcon-AO recall of the matching results was not affected at all, while in the case of matcher OWL-CTXmatch we only removed one correct correspondence. For somematcher++ and HMatch we could increase precision by 8% and 19%. Since we could not compute recall of the matching system due to the missing of a reference mapping, we can make no exact statements about the negative effects on recall for these two matchers.

On the other hand the values for recall of the revision range between 22% and 56%. This means that our debugging method only captures parts of the incorrect correspondences. We have already expected a similar result, since the under-specification the ontologies in terms of missing disjointness statements results in a lack of inconsistency symptoms that are the basis for the repairing algorithm. On the other hand, this means that we can expect much better results for ontologies with carefully specified disjointness statements between concepts.

5.2 Revising with Expert Evaluation

While in the previous section we dealt with the case of a fully automatized revision, in the following we focus on the aspect of supporting mapping revision conducted by

[3]The developers of the matching system referred to as 'somematcher++' below prohibited us to publish results related to their system, therefore we do not use the actual name.

	mapping		revision	
Matching System	**number**	**precision**	**precision**	**recall**
Falcon-AO	246	89%	100%	22 %
OWL-CTXmatch	280	68%	96%	26 %
somematcher++	270	75%	80%	54 %
HMatch	406	57%	78%	56%

Table 1: Experimental Results on the OntoFarm Benchmark. The first column indicates the number of correspondences generated by the matching system, the second the precision of this mapping (figures for recall were not available at the time of the experiment). The last two columns refer to the precision and recall of the revision method, where precision is the fraction of removed correspondences that were actually wrong according to the reference mapping and recall is the fraction of wrong correspondences according to the reference mapping that our method removed.

a domain expert. Without reasoning support the expert has to evaluate each bridge rule in mapping $\mathfrak{B}$ iteratively. In the following we measure the effort of an evaluation process in number of evaluation decisions. An evaluation decision is defined to be the specification of a successor e' of the correct evaluation function e such that $|e'(\mathfrak{B}, unknown)| + 1 = |e(\mathfrak{B}, unknown)|$. The effort for evaluating mapping $\mathfrak{B}$ without support will thus be $|\mathfrak{B}|$. In this section we measure how many evaluation decisions can be skipped by extending the evaluation function after each manual evaluation decision.

We selected four of the ontoFarm ontologies and automatically generated mappings between all pairs of ontologies by applying the matching system CtxMatch [3]. In contrast to the majority of existing systems limited to discovery of "$\equiv$" correspondences, CtxMatch is additionally capable of finding "$\sqsubseteq$", "$\sqsupseteq$" relations. This is more adequate for many applications but makes the manual revision even more time-consuming, because normally the system finds more correspondences than other systems.

For all pairs of ontologies $\langle \mathcal{T}_i, \mathcal{T}_j \rangle$ with $\mathcal{T}_i \neq \mathcal{T}_j \in \{$CMT,CRS,PCS,CONFTOOL$\}$ we built the distributed terminology $\mathfrak{T} = \langle \{\mathcal{T}_i, \mathcal{T}_j\}, \{\mathfrak{B}_{ij}\} \rangle$ where $\mathfrak{B}_{ij}$ is the mapping generated by the CtxMatch matching system. Then we proceeded as follows:

1. Init the counter $m \leftarrow 0$ of manual evaluation decisions.
2. Evaluate the first unevaluated bridge rule $b \in \mathfrak{B}_{ij}$ and set $m \leftarrow m + 1$.
3. Recompute $e \leftarrow$EXTENDEVALUATION$(\mathfrak{T}, e, i, j)$.
4. If $e(\mathfrak{B}_{ij}, unknown) \neq \emptyset$ continue with step 2.

This procedure ends when every bridge rule has been manually or automatically evaluated. While m counts the number of manual evaluation decision, $(|\mathfrak{B}_{ij}| - m)/\,|\mathfrak{B}_{ij}|$ measures the fraction of bridge rules evaluated manually. In addition, we also counted the number of bridge rules that have been evaluated as *correct* by entailment as well

		CMT	CRS	PCS	CONFTOOL
CMT	*size*	-	53 $\rightsquigarrow$ 44		48 $\rightsquigarrow$ 32
	random order		**56.6%** (23/0)	n.a.	**60.4%** (19/0)
	impact order		**35.8%** (33/1)		**39.6%** (28/1)
CRS	*size*	53 $\rightsquigarrow$ 41	-	38 $\rightsquigarrow$ 29	80 $\rightsquigarrow$ 38
	random order	**54.7%** (23/1)		**60.5%** (15/0)	**65%** (18/10)
	impact order	**41.5%** (29/2)		**52.6%** (18/0)	**22.5%** (36/26)
PCS	*size*	73 $\rightsquigarrow$ 63	38 $\rightsquigarrow$ 30	-	45 $\rightsquigarrow$ 23
	random order	**41.1%** (43/0)	**60.5%** (15/0)		**73.3%** (12/0)
	impact order	**27.4%** (53/0)	**52.6%** (18/0)		**55.6%** (19/1)
CONFTOOL	*size*	48 $\rightsquigarrow$ 32	80 $\rightsquigarrow$ 36	45 $\rightsquigarrow$ 23	-
	random order	**60.4%** (19/0)	**68.8%** (18/7)	**73.3%** (12/0)	
	impact order	**43.8%** (27/0)	**40%** (36/12)	**57.8%** (19/0)	

Table 2: Experimental results for supporting manual evaluation. The first row in each cell represents $|\mathfrak{B}| \rightsquigarrow |e(\mathfrak{B}, \mathit{correct})|$ for the finally obtained evaluation function e. The second and third row distinguish between iterating over different orderings of the input mapping. They present the fraction of bridge rules that had to be evaluated. In parentheses you find the number of bridge rules automatically selected due to entailment and the number of bridge rules unselected due to inconsistency.

as the number of bridge rules that have been evaluated as $\mathit{incorrect}$ by inconsistency checking.

In a first series of experiments we ordered the bridge rules in a random way.[4] The results for these experiments are presented in the rows headed with random order in table 2. The fraction of bridge rules that had to be evaluated manually ranges from 41.1% to 73.3%. Aggregating over all pairs of ontologies, we measured that only 60.8% of all bridge rules had to be evaluated instead of evaluating 100% in a scenario without revision support. Notice that most parts of the extension are based on entailment, while reasoning with inconsistencies has only limited effects. As we already mentioned, the ontologies in our test set are underspecified with respect to disjointness between concepts. This prevents some mappings to become inconsistent even though incorrect correspondences are involved.

Even though these results show the benefit of our approach, there is still room for improvement by ordering the bridge rules of the input mapping in a proper way. The following example describes the effects of different orderings.

Example 3 *The example from the introduction also nicely shows the importance of a good ordering. Given the input mapping* $\mathfrak{B}_{12} = \{b_{23}, b_{24}, b_{25}\}$ *from example 1.*

- b_{23} *and* b_{25} *are correct and* b_{24} *is incorrect,*
- $\{b_{23}\}$ *entails* b_{25}*,*

[4] More precisely, to make the results reproducible we ordered the bridge rules lexicographical with respect to the concepts matched by the bridge rule.

- *and* $\{b_{23}, b_{24}\}$ *is inconsistent.*

On the one hand, if we first manually evaluate $e(b_{23}) = correct$ *the extension of* e *will be a complete evaluation function with* $e(\mathfrak{B}, unknown) = \emptyset$*. On the other hand, if we evaluate* $\mathfrak{B}$ *in the order of* $\langle b_{24}, b_{25}, b_{23}\rangle$ *the extension function cannot be extended at all.*

Example 3 shows that we have to find an appropriate order for a given input mapping to exploit our approach to its full extent. To determine such an order we define the notion of the potential impact of a bridge rule, formally introduced in definition 11. Given a bridge rule b from $\mathcal{T}_1$ to $\mathcal{T}_2$ the potential impact counts the number of bridge rules b' that can be entailed from $\{b\}$ as well as the number of bridge rules such that $\{b, b'\}$ is inconsistent, where $b' \in \mathfrak{B}_{full}$ and $\mathfrak{B}_{full}$ is defined to be the set of all combinatorial possibilities for matching concepts from $\mathcal{T}_1$ to $\mathcal{T}_2$. Notice that this characteristic is only a rough approximation of a bridge rules' real impact, because it abstracts from complex interactions between more than two bride rules.

Definition 11 (Potential impact of a bridge rule) *The potential impact of a bridge rule from* $\mathcal{T}_1$ *to* $\mathcal{T}_2$ *is defined as*

$$imp(\mathcal{T}_1, \mathcal{T}_2, 1{:}C \xrightarrow{R} 2{:}D) \mapsto \begin{cases} sub(\mathcal{T}_1, C) \cdot (super(\mathcal{T}_2, D) + dis(\mathcal{T}_2, D)) & \text{if } R = \sqsubseteq \\ super(\mathcal{T}_1, C) \cdot (sub(\mathcal{T}_2, D) + dis(\mathcal{T}_2, D)) & \text{if } R = \sqsupseteq \\ imp(\mathcal{T}_1, \mathcal{T}_2, 1{:}C \xrightarrow{\sqsubseteq} 2{:}D) + imp(\mathcal{T}_1, \mathcal{T}_2, 1{:}C \xrightarrow{\sqsupseteq} 2{:}D)) & \text{if } R = \equiv \end{cases}$$

where $sub(\mathcal{T}, C)$ *returns the number of all subclasses of concept* C *in* $\mathcal{T}$*,* $super(\mathcal{T}, C)$ *returns the number of all superclasses of concept* C *in* $\mathcal{T}$*, and* $dis(\mathcal{T}, C)$ *returns the number of all classes that are disjoint with* C*.*

For a second series of experiments we ordered the bridge rules descending due to their potential impact. The results are also presented in table 2 in the rows headed with impact order. The effects confirm with our theoretical expectations. The number of entailment propagations as well as the number of inconsistency propagations could be increased by a significant degree. We reduced the effort of manual evaluation to the range from 22.5% to 57.8%. In average we now have to evaluate only 40.4% of the input mapping. This means that a domain expert has to evaluate less than every second bridge rule of a mapping in average.

6 Related Work

There are at least two areas of research that have a direct connection to the work reported in this paper. The first area is concerned with the revision of ontologies, the second with the analysis and generation of semantic relations between different ontologies.

In a recent paper, Haase and Qi compare different approaches for resolving inconsistencies in Description Logic Ontologies [9]. Amongst the approaches reviewed in the paper, there is a group of methods that extend and adapt classical methods from belief revision, in particular the AGM theory to the case of description logics. While [6] only considers formal properties of ontology revision in terms of a modification

of the AGM postulates and the formal properties of revision operators, [16] proposes a revision operator based on weakening the model by introducing exceptions. In [28] Wassermann argues that diagnostic reasoning can be seen as a special kind of believe revision. Recently a number of approaches for diagnosing and repairing inconsistencies and incoherence in ontologies have been proposed. These approaches are similar to our work in the sense that they apply diagnostic reasoning to the problem of belief revision, however none of these approaches addresses the revision of mappings between ontologies.

In [13] Parsia and other discuss different approaches to ontology diagnosis and debugging. In particular, they distinguish glass-box techniques that analyze the tableaux proof to find causes of inconsistencies and black-box techniques that use the reasoning algorithm as a black box and try to detect inconsistencies by successively asking questions to the reasoner. Our approach for detecting inconsistencies in mappings can be seen as a black-box approach in this context.

Schlobach and Cornet [20] investigate the problem of computing minimal conflict sets (called MUPS) for $\mathcal{ALC}$ terminologies as a basis for diagnostic reasoning. In follow up work [19] Schlobach investigates a glass-box approach that uses of the hitting set algorithm as a basis for computing diagnoses. Several extensions of the basic method by Schlobach have been developed. In [10] the authors extend the hitting set based diagnosis algorithm for ontologies to the language $\mathcal{SHOIN}$ thereby covering most of the expressive power of OWL-DL. The authors of [7] propose a black-box method based on the hitting set algorithm that does not require a modification of the reasoning algorithm. The work of Schlobach revealed that the standard hitting set algorithm is not the best choice with respect to diagnosing ontological knowledge. In our work, we adopted the idea of computing minimal conflict sets and of replacing the standard diagnosis algorithm by a specialized diagnosis strategy. All of the approaches mentioned above solely work on a single ontology. A first attempt towards extending the approaches mentioned above towards mapped ontologies is made in the NeOn Project [14]. The resulting method has been tested on realistic ontologies [15] one including mappings between different ontologies. The approach, however, does not distinguish between mappings and local axioms. Our work focusses on mappings which allows us to use special heuristics not applicable in the general case, e.g. by relying on the confidence values provided by the matching systems.

Wang and Xu [27] report some work that explicitly addresses the problem of debugging ontology mappings. They propose a number of heuristics for identifying and repairing inconsistent mappings. Most of the heuristics proposed for detecting inconsistent mappings are special cases of the general framework of inconsistency detection described in this paper. Similar heuristics for identifying conflicts are used by state-of the art matching systems for eliminating mapping hypotheses. None of these approaches, however, is based on a sound model-theoretic notion of unsatisfiability.

A notable exception is the work on semantic matching carried out at the University of Trento [8]. Their approach uses propositional logical reasoning about concept labels for identifying mappings. This strategy ensures that no inconsistent mappings are generated in the first place. The approach has been extended to more expressive ontologies in [23]. A drawback of this method compared to our approach is the fact that the use of logical reasoning for proving semantic relations is often too restrictive and will miss

many correct matches. Our approach addresses this problem by first allowing the use of rather weak heuristics for generating hypotheses and only using reasoning a posteriori to filter out obviously incorrect hypotheses. This will in general lead to a higher recall.

7 Conclusions and Future Work

In this paper, we have argued for the need of providing reasoning support for manual mapping revision to deal with the inherent complexity of the problem. We proposed methods for dealing with the revision problem that can either be applied directly to a proposed mappings to find and remove inconsistencies or to support the process of manual mapping revision by a human expert. We have implemented a graphical tool for manual revision that includes all of the methods mentioned in section 4 and used it to perform some experiments reported in section 5. The experiments show that the revision methods proposed improve the revision process both in terms of the quality of mappings (in particular the precision is enhanced significantly) and in terms of the human effort in terms of correspondences that have to be evaluated.

The problem of mapping revision addressed in this paper can be seen as a special case of the general problem of ontology or knowledge base revision. This allowed us to build upon some general principles of knowledge revision that have been described in the literature. In particular, the use of diagnostic reasoning to implement the revision process as proposed in [28] has been adopted. Further, our method for computing diagnosis is similar to existing work on ontology diagnosis and relies on the notion of conflict sets as minimal sets of axioms that cause an inconsistency. As we have seen, the special needs of mapping revision also required us to abandon some of the principles normally used in the area of knowledge revision. In particular, the use of the minimality criterion for deciding which axioms to remove from a conflicting set does not perform well in the context of mapping revision (compare example 2). For us this meant that we could not use standard methods for computing diagnosis (in particular the hitting set algorithm [17]). Instead we relied on a greedy strategy for deciding which correspondences to remove which turned out to perform quite well in many cases.

There are a number of open problems that need to be addressed in future work. Some of these problem have already been mentioned in the paper. One is the problem of underspecified ontologies. In particular, the detection of inconsistencies in the mappings relies on the presence of correct disjointness axioms in the mapped ontologies. In practice these axioms are often not available. There are several ways we could deal with this problem. One is to work with the assumption that sibling-concepts are always disjoint and adding the corresponding axioms to the ontologies. This has already successfully been done in the context of revising ontologies [18]. Another option is to apply ontology learning techniques to automatically add missing disjointness statements. A corresponding approach has recently been proposed [26]. Another potential problem is the complexity of the reasoning problems involved. It has been shown that subsumption reasoning in DDL is NEXP-Time-Complete [24]. As our revision method makes extensive use of this reasoning service, we cannot hope for efficiency as long

as we stick to the requirement that the result of the revision step has to be consistent and closed under deduction. Recently, we have explored efficient approximations of these reasoning services that only require to classify the ontologies once and then use correct but incomplete heuristics for checking consistency [11]. So far, we have found only a very few examples where this approximate method fails to detect all inconsistencies. Finally, the method presented in this paper is only complete with respect to mappings between concepts. Many matching systems, however, also propose correspondences between relations in different ontologies. Extending our approach to these correspondences as well will be one of the next steps.

Despite these shortcomings, the general idea of logic-based mapping revision has a lot of benefits also for the matching process. In recent work, we have used a variation of the revision method proposed here to filter out results early in the matching process. Instead of the greedy approach used here, we combined logical reasoning with a full-fledged optimization algorithm. With this combination we where able to increase the quality of matching results significantly. Exploiting this possibility to increase the quality of matching results thus easing the evaluation will also be investigated in more details in future work.

Acknowledgement The work has been partially supported by the German Science Foundation (DFG) in the Emmy Noether Programme under contract STU 266/3-1.

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing Ontologies. *Journal on Web Semantics*, 1(4):325–343, 2004.

[3] P. Bouquet, L. Serafini, and S. Zanobini. Peer-to-peer semantic coordination. *Journal of Web Semantics*, 2(1):81 97, December 2004.

[4] Maria Garcia de la Banda, Peter Stuckey, and Jeremy Wazny. Finding all minimal unsatisfiable subsets. In *Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declaritive programming*. ACM Press, 2003.

[5] Jerome Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer Verlag, 2007.

[6] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. On applying the agm theory to dls and owl. In Yolanda Gil, Enrico Motta, Richard Benjamins, and Mark Musen, editors, *roceedings of the 4th International Semantic Web Conference (ISWC-05)*, volume 3729 of *Lecture Notes in Computer Science*, pages 216–231, 2005.

[7] Gerhard Friedrich and Kostyantyn Shchekotykhin. A general diagnosis method for ontologies. In *Proceedings of 4th International Conference on Semantic Web (ISWC05)*, pages 232–246, Galway, Ireland, 2005.

[8] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics*, IX, 2007.

[9] Peter Haase and Guilin Qi. An analysis of approaches to resolving inconsistencies in dl-based ontologies. In *Proceedings of ESWC 2007 Workshop on Ontology Dynamics*, 2007.

[10] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. Repairing unsatisfiable concepts in owl ontologies. In *Proceedings of the 3rd European Semantic Web Conference*, pages 170–184, Budva, Montenegro, 2006.

[11] Christian Meilicke and Heiner Stuckenschmidt. Applying logical constraints to ontology matching. In Joachim Hertzberg, Michael Beetz, and Roman Englert, editors, *Proceedings of the 30th German Conference on Artificial Intelligence*, number 4667 in Lecture Notes in Artificial Intelligence, pages 99–113. Springer, September 2007.

[12] Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin. Repairing ontology mappings. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, Vancouver, Canada, 2007.

[13] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging owl ontologies. In *Proceedings of the 14th international World Wide Web Conference*, page 633640, Chiba, Japan, 2005.

[14] Guilin Qi and Peter Haase. Consistency model for networked ontologies. NEON Deliverable D1.2.1, The NEON Project, 2007.

[15] Guilin Qi and Peter Haase. Consistency models for networked ontologies evaluation. NEON Deliverable D1.2.2, The NEON Project, 2007.

[16] Guilin Qi, Weiru Liu, and David Bell. A revision-based approach to handling inconsistency in description logics. In *Proceedings of 11th International workshop on Non-Monotonic Reasoning (NMR06)*, pages 124–132, 2006.

[17] Ray Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[18] S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proceedings of ESWC 2005*, 2005.

[19] Stefan Schlobach. Diagnosing terminologies. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, page 670675, 2005.

[20] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, page 355362, Acapulco, Mexico, 2003.

[21] 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.

[22] Luciano Serafini and Andrei Tamilin. DRAGO: Distributed reasoning architecture for the semantic web. In *Proceedings of the Second European Semantic Web Conference (ESWC'05)*, 2005.

[23] Luciano Serafini, Stefano Zanobini, Simone Sceffer, and Paolo Bouquet. Matching hierarchical classifications with attributes. In *Proceedings of the 3rd European semantic web Conference ESWC 2006*, pages 4–18, 2006.

[24] Heiner Stuckenschmidt and Michel Klein. Reasoning and change management in modular ontologies. *Data and Knowledge Engineering*, 63(2):200–223, November 2007.

[25] Ondrej Svab, Svatek Vojtech, Petr Berka, Dusan Rak, and Petr Tomasek. Ontofarm: Towards an experimental collection of parallel ontologies. In *Poster Proceedings of the International Semantic Web Conference 2005*, 2005.

[26] Johanna Völker, Denny Vrandecic, York Sure, and Andreas Hotho. Learning disjointness. In *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*. Springer, 2007.

[27] Peng Wang and Baowen Xu. Debugging ontology mapping: A static method. *Computing and Informatics*, 22:1001–1015, 2007.

[28] Renata Wassermann. An algorithm for belief revision. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*. Morgan Kaufmann, 2000.