Repairing Ontology Mappings

C. Meilicke and **H. Stuckenschmidt**
KR and KM Research Group
University of Mannheim
A5, 6 68159 Mannheim, Germany
{christian, heiner}@informatik.uni-mannheim.de

Andrei Tamilin
ITC-irst
Via Sommarive, 18
38050 Povo (Trento) Italy
tamilin@itc.it

Abstract

Automatically discovering semantic relations between ontologies is an important task with respect to overcoming semantic heterogeneity on the semantic web. Existing ontology matching systems, however, often produce erroneous mappings. In this paper, we address the problem of errors in mappings by proposing a completely automatic debugging method for ontology mappings. The method uses logical reasoning to discover and repair logical inconsistencies caused by erroneous mappings. We describe the debugging method and report experiments on mappings submitted to the ontology alignment evaluation challenge that show that the proposed method actually improves mappings created by different matching systems without any human intervention.

Motivation

Recently, a number of heuristic methods for matching concepts from different ontologies have been proposed. These methods rely on linguistic and structural criteria. Evaluation studies have shown that existing methods often trade off precision and recall. The resulting mapping either contains a fair amount of errors or only covers a small part of the ontologies involved (Euzenat *et al.* 2006; Euzenat, Stuckenschmidt, & Yatskevich 2005). This means that automatically created mappings often contain errors in the form of mappings that do not reflect the semantic relation between concepts. Currently, the only way to deal with this problem without manual intervention is to sacrifice recall by choosing a matching approach that optimizes for precision. Our goal is to solve the problem of errors in automatically created mappings by automatically identifying and repairing errors in mappings.

Problem Statement

The problem of ontology matching can be defined in the following way (Euzenat & Shvaiko 2007). Ontologies are theories encoded in a certain language L. In this work, we assume that ontologies are encoded in OWL-DL without nominals. For each ontology $\mathcal{T}$ in language L there is a function $Q(\mathcal{T})$ that defines matchable elements of the ontology. Given two ontologies $\mathcal{T}$ and $\mathcal{T}'$ the task of matching is now to determine correspondences between the matchable elements in the two ontologies. Correspondences are 5-tuples $\langle id, e, e', r, n\rangle$ such that

- id is a unique identifier for referring to the correspondence
- $e \in Q(\mathcal{T}), e' \in Q'(\mathcal{T}')$ are matchable elements from the two ontologies
- $r \in R$ is a semantic relation. In this work, we only consider cases where $R = \{\equiv, \sqsubseteq, \sqsupseteq\}$
- n is a confidence value describing the confidence in the correctness of the correspondency.

The problem we address in this paper is now, given a set of correspondences C between two ontologies $\mathcal{T}$ and $\mathcal{T}'$ and a (possibly unknown) set of correspondences G which is a reference mapping for the ontologies involved, to determine $C \cap G$. As G is often not accessible or not even known, we reformulate the problem in terms of a partitioning of C into correct and incorrect correspondences with respect to the reference mapping:

- $C^+ =_{def} C \cap G$
- $C^- =_{def} C - G$

Based on this formalization, the task of repairing a set of correspondences is to determine for each correspondence $c \in C$ whether it belongs to C^+ or C^-. Repairing C then corresponds to deleting all correspondences that are in C^- from C.

Approach and Contributions

The approach taken in this work is to interpret the problem defined above as a diagnosis task. For this purpose, we formalize correspondences in distributed description logics and analyze the impact of the mapping on the ontologies it connects. The basic assumption is that a mapping that correctly states the semantic relations between ontologies should not cause inconsistencies in any of the ontologies. The encoding in distributed description logics allows us to detect these inconsistencies which are treated as symptoms caused by an incorrect mapping. We then compute sets of correspondences that jointly cause a symptom and repair each symptom by removing correspondences from these sets. The set of correspondences remaining after this process can be regarded as an approximation of C^+. The concrete contributions of this work are:

Copyright © 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

- We propose a method for automatically repairing mappings based on terminological reasoning that goes beyond structural heuristics used in existing matchers.
- We implemented this method using the DRAGO reasoning system for distributed description logics.
- We performed experiments evaluating the performance of the method applied on the result of different matching systems.

The paper is organized as follows. First, we briefly summarize distributed description logics as a basis for formalizing ontology mappings and explain how correspondences can be encoded in this formalism. We then formally introduce the mapping property of inconsistency. Based on this property the formulation of mapping repair as a diagnosis task takes place. Finally, we describe the experiments we conducted and their results followed by a general discussion of the approach and future work.

Formalizing Mappings

In the following it is assumed that the reader is familiar with description logics. An introduction can be found in (Baader *et al.* 2003). Distributed description logics, as described by Serafini and Tamilin in (Serafini & Tamilin 2005), can be understood as a framework for formalization of multiple ontologies pairwise linked by directed semantic mappings. In this context a pair of ontologies $\mathbb{T}$ and associated mappings $\mathbb{M}$ is called a distributed ontology $\mathfrak{T} = \langle \mathbb{T}, \mathbb{M} \rangle$.

Let I be set of indices. Then $\mathbb{T} = \{\mathcal{T}_i\}_{i \in I}$ denotes the set of all ontologies in $\mathfrak{T}$ and $\mathcal{T}_i$ with $i \in I$ denotes the i-th ontology of $\mathfrak{T}$. Each ontology $\mathcal{T}_i$ 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 ontology $\mathcal{T}_i$, the index of the ontology 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 two types of bridge rules are known in distributed description logics.

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

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 ontology $\mathcal{T}_i$. 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 maps 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 ontologies $\langle \mathcal{T}_i, \mathcal{T}_j \rangle_{i \neq j}$ 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 ontology $\mathfrak{T}$ 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 $\mathcal{M}_{ij}$

Due to the introduction of bridge rules it is possible to transfer knowledge between different ontologies that changes subsumption relations in the target ontology. In particular, the following inference rule can be used to infer new subsumption relations across ontologies:

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

It has been shown that this general propagation rule completely describes reasoning in DDLs that goes beyond well known methods for reasoning in Description Logics. To be more specific, adding the inference rule in equation 1 to existing tableaux reasoning methods leads to a correct and complete method for reasoning in DDLs. A corresponding result using a fixpoint operator is given in (Serafini, Borgida, & Tamilin. 2005).

We use the framework of distributed description logics to formalize correspondences generated by automatic matching tools. In particular, each correspondence (id, e, e', r, n) is translated into a set of bridge rules using a translation function t in the following way:

$$t(\langle id, e, e', \sqsubseteq, n \rangle) = \{1:e \xrightarrow{\sqsupseteq} 2:e', 2:e' \xrightarrow{\sqsubseteq} 1:e\}$$

$$t(\langle id, e, e', \sqsupseteq, n \rangle) = \{1:e \xrightarrow{\sqsubseteq} 2:e', 2:e' \xrightarrow{\sqsupseteq} 1:e\}$$

Equivalence correspondences are interpreted as a pair of inclusion correspondences which are treated individually in our experiments. If a bridge rules causes a problem, we can consider this problem to be caused by the correspondence the rule was translated from.

Example 1 *Consider two ontologies in the domain of conference management systems, the same domain we used in our experiments. For each ontology consider a single axiom, namely:*

$$i : Author \sqsubseteq Person$$
$$j : Person \sqsubseteq \neg Authorization$$

For the sake of simplicity, we assume that a simple string matching method that computes a similarity value that denotes the relative size of the common substring. Correspondences are created based on a threshold for this value that

we assume to be 1/3. Applying this method to the example will result in the following correspondences:

$$\langle 42, i : Person, j : Person, \equiv, 1.00\rangle$$
$$\langle 43, i : Author, j : Authorization, \sqsupseteq, 0.46\rangle$$

As mentioned above, we treat the correspondence 42 as two subsumption correspondences (42i and 42o), i.e.

$$\begin{aligned} C \;=\; & \{\langle 42i, i : Person, j : Person, \sqsubseteq, 1.00\rangle, && (2)\\ & \langle 42o, i : Person, j : Person, \sqsupseteq, 1.00\rangle, && (3)\\ & \langle 43, i : Author, j : Authorization, \sqsupseteq, 0.46\rangle\} && (4)\end{aligned}$$

Applying the transformation function to the first correspondence yields the following set of bridge rules:

$$\{i : Person \xrightarrow{\sqsubseteq} j : Person, \quad (5)$$
$$i : Person \xleftarrow{\sqsubseteq} j : Person\} \quad (6)$$

In the same way we translate the other correspondences. The union of the resulting sets corresponds to the mapping set $\mathbb{M}$ which in our example consists of six bridge rules. Together with the axioms of the ontologies involved these bridge rules form the basis for deriving new knowledge that helps in the debugging process.

Reasoning and Debugging

The encoding of correspondences in terms of bridge rules in distributed description logics enables us to formally reason about the impact mappings have on the connected ontologies. In the following, we present a number of formal properties that are used in the process of debugging and that can be tested using the DRAGO reasoning system (Serafini & Tamilin 2005).

A mapping of a distributed ontology can be defined as inconsistent with respect to a concept $i : C$ if the additional constraints induced by the mapping have the (unintended) effect of making the locally satisfiable concept $i : C$ distributedly unsatisfiable. If such an effect does not occur the mappings are consistent with respect to $i:C$.

Definition 1 (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 $\mathcal{T}_i$.*

Example 2 *In our example above, the set $\mathbb{M}$ of generated bridge rules contains – amongst others – the following rules:*

$$i{:}Author \xrightarrow{\sqsupseteq} j{:}Authorization$$
$$i{:}Person \xrightarrow{\sqsubseteq} j{:}Person$$

We can see that $\mathbb{M}$ is inconsistent with respect to $Authorization$. This is true because we can derive by distributed reasoning, that $j : Authorization \sqsubseteq Person$ has to hold. At the same time, $Authorization$ and $Person$ are defined as disjoint concepts in ontology $\mathcal{T}_j$. In particular, this makes $Authorization$ unsatisfiable with respect to the global interpretation.

Using the property of consistency one can make statements about differences between the local and the distributed taxonomy of an ontology. In the context of mapping debugging this approach will be extended by comparing the distributed taxonomy to the distributed taxonomy that results from a modified set of bridge rules.

Diagnosis

In the following we rely on the classical definition of diagnosis introduced by Reiter (Reiter 1987). The basic assumption of our approach is that mappings model semantic correspondences between concepts in 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 the context of mapping debugging we regard the correspondences in C to be the set of components to be diagnosed. The distributed ontology $\mathfrak{T}$ consisting of the mapped ontologies and the bridge rules generated by applying t to C 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 of the mapping. In other words, a diagnosis is the minimal set of correspondences Δ such that the mapping $\mathbb{M} - t(\Delta)$ it consistent.

Definition 2 (Diagnosis) *Let $\mathfrak{T} = \langle \mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2\}, \mathbb{M} = t(C)\rangle$ be a distributed ontology. A diagnosis for $\mathfrak{T}$ is defined as the minimal set $\Delta \subseteq C$ such that $\mathbb{M} - t(\Delta)$ is consistent with respect to $\mathcal{T}_1$ and $\mathcal{T}_2$.*

Example 3 *In our example, there are two diagnoses explaining the inconsistency of the mapping. In particular, there are two minimal subsets of C that satisfy the requirements of definition 2, namely $\{i42\}$ and $\{43\}$ because both $\mathbb{M} - t(i42)$ and $\mathbb{M} - t(43)$ are consistent mappings with respect to $\mathcal{T}_i$ and $\mathcal{T}_j$.*

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 $COMP$ which in our case is the number of correspondences generated by the matcher. In order to deal with this problem, we adopt the notion of conflict sets (Reiter 1987) for guiding the search for abnormal correspondences.

Conflict Sets

The notion of a conflict set is central to existing algorithms for computing diagnosis. Reiter defines a conflict set as a subset of the system components that together produce an abnormal behavior. In our case a conflict set is just a subset

of the mapping that is still strong enough to be inconsistent in the sense of definition 1. 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 non 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 fixing one component in the set. This means that the problem of identifying an incorrect bridge rule boils down to computing minimal conflict sets and to decide which of the rules involved is incorrect and should thus be removed from the mapping. A minimal conflict set in the context of mapping debugging can be defined as follows.

Definition 3 (Minimal Conflict Set) *Let* $\mathfrak{T} = \langle \mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2\}, \mathbb{M} \rangle$ *be a distributed ontology. A set of bridge rules* $\mathbb{C} \subseteq \mathbb{M}$ *is a conflict set for a concept* $i{:}C$ *with* $i \in \{1, 2\}$ *iff for* $\mathfrak{T}' = \langle \mathbb{T}, \mathbb{C} \rangle$ *we have* $\mathfrak{T}' \models i{:}C \sqsubseteq \bot$ *and* $\mathcal{T}_i \not\models C \sqsubseteq \bot$. $\mathbb{C}$ *is a minimal conflict set for* $i{:}C$ *iff* $\mathbb{C}$ *is a conflict set for* $i{:}C$ *and iff there exists no* $\mathbb{C}' \subset \mathbb{C}$ *that is also a conflict set for* $i{:}C$.

Note that there can be more than one minimal conflict set for the same concept as we can see from the following example.

Example 4 *The bridge rules listed in example 2 are a minimal conflict set for the concept* $Authorization$. *Together they make* $Authorization$ *inconsistent but any subset is not a conflict set for this concept. If* C *would contain the additional correspondence* $\langle 44, i : Person, j : Authorization, \sqsupseteq, 0.46 \rangle$ *the resulting mapping would also contain another conflict set consisting of the following bridge rules* $i : Person \xrightarrow{\sqsupseteq} j : Authorization$ *and* $i : Person \xrightarrow{\sqsubseteq} j : Person$

After minimal conflict sets have been determined, they can be used to resolve the conflict caused by the specific combination of bridge rules in a conflict set. This can be done by deleting a single bridge rule from the respective conflict set. Since the same symptom can be caused by different minimal conflict sets, every conflict set has to be treated this way until the unwanted effect does not occur any more.

As stated before, the assumption is that if properly designed, the debugging method will delete exactly those bridge rules that represent incorrect correspondences. This implies that the step of selecting bridge rules to be deleted is the crucial one in the debugging process. A naive approach to this problem is the random selection of one of the elements of the conflict set. In classical diagnosis, all conflict sets are computed and the diagnosis is computed from these conflict sets using the hitting set algorithm. For the case of diagnosing mappings this is neither computationally feasible nor does it provide the expected result. In the case of our example, applying the hitting set method to the two existing conflict sets does not lead to the wanted result as the hitting set consists of the bridge rule $i : Person \xrightarrow{\sqsubseteq} j : Person$ which is not the real problem in this case.

An obvious way to improve the approach is to make use of the degree of confidence that is assigned to each correspondence. A low confidence value indicates that the matcher has some doubts that the semantic relation encoded in the correspondence is actually correct. Therefore, rules with a low confidence value can be assumed to be more likely incorrect than rules with a high value.

But since not every matcher evaluates generated correspondences with an expressive confidence value, we needed an alternative measure for deciding which mapping to remove from a conflict set. Therefore, we implemented a simple heuristic that computes the Wordnet distance between the concepts connected by a mapping. In particular, we used the similarity measure proposed by (Seco, Veale, & Hayes 2004). Applying this heuristic makes the approach independent of the ability of the matching systems to compute the confidence for a correspondence. It also turned out that for those matchers that provided a measure of confidence the use of the semantic similarity as a basis for selecting correspondences to be removed outperformed the confidence-based method. The corresponding experiments are omitted here due to lack of space.

The algorithm for mapping repairing has already been described. A formal representation is given in algorithm 1. Note, that the procedure GETLOWESTBRIDGERULE($\mathbb{C}$) returns the bridge rule in $\mathbb{C}$ with the lowest confidence estimation of the correspondence this bridge rule was translated from. The confidence values are computed as described in the last section. For each symptom discovered by algorithm

Algorithm 1

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

```
1: for all C ∈ GETALLCONCEPTS(𝔗, j) do
2:   if T_j ⊭ C ⊑ ⊥ then
3:     while 𝔗 ⊨ j:C ⊑ ⊥ do
4:       ℂ = GETMINIMALCONFLICSET(𝔗, i, j:C)
5:       b = GETLOWESTBRIDGERULE(ℂ)
6:       𝕄 = 𝕄 − {b}
7:     end while
8:   end if
9: end for
10: return 𝕄
```

1 the procedure GETMINIMALCONFLICSET($\mathfrak{T}, i, j : C$) is called (see algoritm 2). This procedure minimizes the mapping from $\mathcal{T}_i$ to $\mathcal{T}_j$ until a minimal conflict set remains.

Experimental Evaluation

The goal of the experiments was to show that our debugging method actually improves the results of automatic matching. In particular, we wanted to show that the benefit gained from identifying wrong mappings is larger than the damage caused by deleting correct mappings. In the following, we first present the data set and the matching systems used in the experiments. We then briefly define the setting of the experiments and discuss the results.

Algorithm 2

GETMINIMALCONFLICSET($\mathfrak{T}, i, j:C$)

1: **for all** $b \in \mathbb{M}$ **do**
2: $\quad \mathbb{M} = \mathbb{M} - \{b\}$
3: $\quad$ **if** $\mathfrak{T} \not\models j:C \sqsubseteq \bot$ **then**
4: $\quad\quad \mathbb{M} = \mathbb{M} \cup \{b\}$
5: $\quad$ **end if**
6: **end for**
7: **return** $\mathbb{M}$

Experimental Setting

We evaluated our method using automatically created mappings between ontologies of the ontoFarm Dataset. It consists of a set of ontologies in the domain of conference organization that has been created by researchers of the Knowledge Engineering Group at the University of Economics Prague (Svab *et al.* 2005)[1]. Further, we used mappings that have been created by the participants of the 2006 Ontology Alignment Evaluation Campaign (Euzenat *et al.* 2006). Six of the participants submitted results on the OntoFarm dataset. We used the results of those four participants that submitted pairwise alignments for all of the ontologies used in the experiments, in particular falcon, coma++, hmatch and OWL-CTXmatch.

In our experiments we considered pairwise mappings produced between the six ontologies presented above for each of the mapping systems. Overall this makes 60 mappings, 15 for each matching method. We manually evaluated all correspondences to determine the set C^- of incorrect correspondences. 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.

We translated all sets of correspondences into distributed description logic by splitting up equivalences into inclusion correspondences and creating two bridge rules for each correspondence as described above. For each pair of ontologies $\mathcal{T}_i$ and $\mathcal{T}_j$, we ran algorithm 1 in both directions. If one of the bridge rules has been removed during the repair process we judged the associated correspondence as incorrect.

For each mapping, we determined the precision of the mapping as well as the precision and recall of the different debugging methods. The corresponding measures are defined as follows:

$$\text{precision of mapping} = \frac{|C^+|}{|C|}$$

$$\text{repair precision} = \frac{\text{removed correspondences in } C^-}{\text{removed correspondences}}$$

$$\text{repair recall} = \frac{\text{removed correspondences in } C^-}{|C^-|}$$

We computed an overall score by taking the average of the values over all mappings in the experiment for each matching system.

Results

Figure 1 summarizes the results with respect to precision of the matching system and precision and recall of the debugging algorithm. The perfection 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. If we would randomly chose four of the coresondence from one of the mapping, e.g created by OWL-CTXmatch, we would remove in average one incorrect and three correct correspondences, while our debugging method would remove three incorrect and only one correct correspondence.

Applying our repairing strategy to the matching systems falcon and coma we were able to increase the precision by 2% and 6%, respectively. In the case of falcon recall of the matching results was not affected at all, while in the case of coma we only removed one correct correspondence. For OWL-CTXmatch 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.

	mapping		**repair**	
Matching System	**number**	**precision**	**precision**	**recall**
falcon	246	89%	100%	22 %
OWL-CTXmatch	270	75%	80%	54 %
coma	280	68%	96%	26 %
hmatch	406	57%	78%	56%

Table 1: Experimental Results on the OntoFarm Benchmark

On the other hand the values for recall of the repairing 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 of mappings and ontologies results in a lack of inconsistency symptoms that are the basis for the repairing algorithm, as argued above.

A potential problem of our approach is the computational complexity of the logical reasoning involved in determining inconsistencies and minimal conflict sets. In our experiments debugging a single mapping between two ontologies took between 4.3 and 19.4 seconds on average with respect to the mappings created by a certain matcher. It turns out that the run-time of the method grows nearly linear with the number of inconsistent concepts as well as linear with the size of the mapping. This also follows from the definition of algorithms 1 and 2. The runtime is also strongly affected by the complexity of distributed reasoning while performing

[1]The ontologies are available at `http://nb.vse.cz/svabo/oaei2006/`

the basic operation of checking $\mathfrak{T} \models j : C \sqsubseteq \bot$ for a certain concept C. Overall, we can say that the runtime for our debugging method is acceptable.

Summary and Conclusions

One problem with using conflict sets is that inconsistency, even in the presence of incorrect correspondences, does not always occur. This might be caused by an underspecification of the ontologies involved or by an underspecification of the correct parts of a mapping. A similar problem has been addressed by Schlobach in the context of debugging Description Logic ontologies (Schlobach 2005). In particular, the author introduces the strong disjointness assumption stating that in a well modeled ontology all direct children of the same parent concept should be regarded as being disjoint. Alternatively, a mapping can be defined as instable with respect to concept $i:C$ if there exists a concept $i:D$ such that $\mathcal{T}_i \not\models C \sqsubseteq D$ and $\mathfrak{T} \models i:C \sqsubseteq i:D$ (Meilicke 2006). Instead of introducing disjointness of sibling concepts it is also possible to regard instability in contrast to inconsistency as symptom for defect mappings. Notice that the set of inconsistency symptoms is a subset of the set of instability symptoms. Further research has to show in how far these approaches can be applied in order to increase recall of our debugging strategy. An interesting theoretical question is about the relation of our work to the theory of belief revision, in particular Investigating the formal properties of our approach to removing mappings based on the theory of contraction functions. As described in (Wassermann 2000) diagnostic reasoning can be used to implement kernel contraction functions for belief revision.

On a broader perspective our experiment showed that there actually is a need for debugging mappings as many existing matchers fail to produce consistent mappings. Some examples of obviously incorrect mappings produced by matching systems in the experiments are the following:

$$\begin{aligned} Document &= Topic \\ Decision &= Location \\ Reception &= Rejection \end{aligned}$$

The real benefit of our method is its ability to also find non-obvious errors in mappings that can only be detected taking the position of the mapped concepts in the concept hierarchy into account. Some examples we found in our experiment:

$$\begin{aligned} Regular_Paper &= Regular \\ Reviewing_event &= review \\ Main_office &\sqsubseteq Location \end{aligned}$$

In the case of the first correspondence, $Regular$ actually denotes the regular participation fee as opposed to the early registration. The error in the second correspondence is caused by the fact that $Reviewing_event$ represents the process of reviewing whereas $review$ denotes the review document as such. The last correspondence is not correct, because the concept $Main_office$ actually represents the main office as an organizational unit rather than a location.

In summary, we have shown that the methods for debugging ontology mappings is a promising approach for improving the quality of automatically created ontology mappings. As the approach works on a standardized representation of mappings, it is independent of the actual method used to create the mappings. This also means that in principle the method can also be applied to manually created mappings. In summary we can say that our method provides suitable functionality for improving matching systems and editors for networked ontologies.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press.

Euzenat, J., and Shvaiko, P. 2007. *Ontology Matching*. Springer Verlag. To appear.

Euzenat, J.; Mochol, M.; Shvaiko, P.; Stuckenschmidt, H.; Svab, O.; Svatek, V.; van Hage, W. R.; and Yatskevich, M. 2006. First results of the ontology alignment evaluation initiative 2006. In Benjamins, R.; Euzenat, J.; Noy, N.; Shvaiko, P.; Stuckenschmidt, H.; and Uschold, M., eds., *Proceedings of the ISWC 2006 Workshop on Ontology Matching*.

Euzenat, J.; Stuckenschmidt, H.; and Yatskevich, M. 2005. Introduction to the ontology alignment evaluation 2005. In *Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies*.

Meilicke, C. 2006. Reasoning about ontology mappings in distributed description logics. Bachelor Thesis, University of Mannheim.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57–95.

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

Seco, N.; Veale, T.; and Hayes, J. 2004. An intrinsic information content metric for semantic similarity in wordnet. In *Proceedings of ECAI'2004, the 16th European Conference on Artificial Intelligence*.

Serafini, L., and Tamilin, A. 2005. DRAGO: Distributed reasoning architecture for the semantic web. In *Proceedings of the Second European Semantic Web Conference (ESWC'05)*. Springer-Verlag.

Serafini, L.; Borgida, A.; and Tamilin., A. 2005. Aspects of distributed and modular ontology reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI-05*.

Svab, O.; Vojtech, S.; Berka, P.; Rak, D.; and Tomasek, P. 2005. Ontofarm: Towards an experimental collection of parallel ontologies. In *Poster Proceedings of the International Semantic Web Conference 2005*.

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