

Supporting Manual Mapping Revision using Logical Reasoning

Christian Meilicke and Heiner Stuckenschmidt

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

Andrei Tamilin

Data & Knowledge Management Unit
Fondazione Bruno Kessler-IRST
Via Sommarive 18, 38100 Povo di Trento, Italy
tamilin@fbk.eu

Abstract

Finding correct semantic correspondences between ontologies is one of the most challenging problems in the area of semantic web technologies. Experiences with benchmarking matching systems revealed that even the manual revision of automatically generated mappings is a very difficult problem because it has to take the semantics of the ontologies as well as interactions between correspondences 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. We show that the use of these reasoning methods significantly reduces the effort of mapping revision in terms of the number of decisions that have to be made by the expert.

Motivation

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'. Recently, a number of automatic and semi-automatic tools for generating hypotheses about semantic correspondences have been developed (see (Euzenat & Shvaiko 2007) 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 a matching system is often inevitable.

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

- The correctness of mappings depends on the semantics of the ontologies. Therefore, mapping revision requires to completely consider the ontologies linked by the mapping. This makes some form of logical reasoning indispensable which is almost impossible to do manually due to the size and complexity of the ontologies.
- Individual decisions about the correctness of a suggested semantic relation can have an influence on past and fu-

ture 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:

1:Abstract equivalent to *2:Abstract* (1)

1:Document equivalent to *2:Document* (2)

1:Document broader than *2:Review* (3)

At first glance all of these relations look correct. Taking the whole ontologies into account, however, 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). Further we might know that *1:Abstract* is a subclass of *1:Document* and *2:Review* is a subclass of *2:Document*. Taking into account these definitions we can deduce 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.

In this paper, we extend and modify the work on fully automatic debugging of ontology mappings proposed in (Meilicke, Stuckenschmidt, & Tamilin 2007) to the case where the revision of the mapping is done by a human expert. We modify the methods in such a way that they support the human evaluator by computing the implications of his decisions to automatically decide whether other semantic relations in the mapping have to be correct or incorrect. We show that the use of automatic reasoning can reduce the effort of manual evaluation by up to 70%. In the next section we present a formal model of mapping revision as a framework for applying logical reasoning. Afterwards we suggest reasoning methods to support the process of mapping revision. Finally, we present results of applying these methods to real data.

A Formal Model of Mapping Revision

Suppose there are two ontologies, \mathcal{O}_1 and \mathcal{O}_2 , describing the same or largely overlapping domains of interest. According to Euzenat and Shvaiko (Euzenat & Shvaiko 2007), correspondences between elements of these ontologies 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 $Q(\mathcal{O}_2)$. 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 $\langle D, \leq \rangle$.*

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 $\{\equiv, \sqsubseteq, \supseteq\}$. In other words, we only focus on equivalence and subsumption correspondences between concepts. 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 $D = [0.0, 1.0]$. The confidence value n can be seen as a measure of trust in the fact that the correspondence holds.

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 $Q(\mathcal{O}_2)$. \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 $e' \in Q(\mathcal{O}_2)$.*

Let \mathcal{M} be an automatically generated mapping between two ontologies \mathcal{O}_1 and \mathcal{O}_2 . We have already argued that \mathcal{M} will most likely contain some erroneous correspondences. Thus, a domain expert will have to revise the mapping to ensure the quality of the integration. For each correspondence in \mathcal{M} the expert evaluator has to choose between one of the alternatives *correct* and *incorrect*. By default, each correspondence is implicitly evaluated as *unknown* as long as no positive or negative evaluation is available. For each point in time the progress of a revision process can be modeled as a function e that assigns to each correspondence of a given mapping a value from the set $\{\text{correct}, \text{incorrect}, \text{unknown}\}$.

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

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

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 $v \in \{\text{correct}, \text{incorrect}, \text{unknown}\}$.

Mapping revision is a sequential process that starts with $e(\mathcal{M}, \text{unknown}) = \mathcal{M}$ where no correspondence has been evaluated. Then the expert evaluator iteratively evaluates one by one correspondences in the mapping. When each of the correspondences is evaluated and $e(\mathcal{M}, \text{unknown}) = \emptyset$ the revision is completed. In order to model such a stepwise revision process we further introduce the notion of a successor of an evaluation function e .

Definition 4 (Successor Evaluation) *Given an evaluation e , an evaluation e' is a successor of e iff $e(\mathcal{M}, \text{correct}) \subseteq e'(\mathcal{M}, \text{correct})$, $e(\mathcal{M}, \text{incorrect}) \subseteq e'(\mathcal{M}, \text{incorrect})$ and $e(\mathcal{M}, \text{unknown}) \supset e'(\mathcal{M}, \text{unknown})$. A successor e' of e is a direct successor of e iff $|e(\mathcal{M}, \text{unknown})| - 1 = |e'(\mathcal{M}, \text{unknown})|$.*

Without additional reasoning support the process of mapping revision consists of $|\mathcal{M}|$ steps from an evaluation function to its direct successor where each step is based on a manual decision of an evaluator.

We already argued that it is possible to make use of previous decisions to automatically derive that certain correspondences in $e(\mathcal{M}, \text{unknown})$ have to be evaluated as *correct* or *incorrect*. Obviously, such a reasoning strategy would decrease the effort of manual evaluation. By means of logical reasoning it will thus be possible to extend an evaluation function to a successor based on the information encoded in e and the ontologies \mathcal{O}_1 and \mathcal{O}_2 without the need of manual intervention. Therefore, we introduce the notion of an extension function as follows.

Definition 5 (Extension) *Given an evaluation function e , a function $\text{ext}(\mathcal{M}, \mathcal{O}_1, \mathcal{O}_2, e) = e'$ is an extension function iff*

- e' is a successor of or equal to e ,
- $e'(c) = \text{correct}$ iff the mapping $e(\mathcal{M}, \text{correct})$ entails c with respect to \mathcal{O}_1 and \mathcal{O}_2 ,
- $e'(c) = \text{incorrect}$ iff $\{c\} \cup e(\mathcal{M}, \text{correct})$ is an inconsistent mapping with respect to \mathcal{O}_1 and \mathcal{O}_2 .

In definition 5 we are using the notion of entailment and consistency. Even though we have an intuitive understanding of both notions with respect to description logics, so far we neglected to give a precise definition of entailment and consistency for correspondences and mappings. In the next section we present distributed description logics as an appropriate framework for modeling ontologies linked by mappings. Based on this formalization we will define the notion of entailment and consistency as well as describe an algorithm to compute the extension of an evaluation function.

Reasoning Support

Distributed description logics (DDL), 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 DDL a collection of T-Boxes and bridge rules between them forms a distributed T-Box $\mathcal{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathcal{B}_{ij}\}_{i \neq j \in I} \rangle$.

With respect to the problem of revising a mapping between two ontologies \mathcal{O}_1 and \mathcal{O}_2 the set of indices I is defined as $\{1, 2\}$ where \mathcal{T}_i denotes the T-Box of ontology \mathcal{O}_i . A T-Box 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 \mathcal{T}_i , the index i is used in front of the concept, for example $i:C$.

The bridge rules in the set \mathfrak{B}_{ij} establish semantic relations from \mathcal{T}_i to \mathcal{T}_j . Every bridge rule in \mathfrak{B}_{ij} has a certain type and connects a concept from \mathcal{T}_i to a concept from \mathcal{T}_j . The following three types of bridge rules are known in DDL:

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

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

Obviously, the role of bridge rules in DDL captures our intuitive understanding of correspondences. Thus, we can respectively map the correspondences of a mapping \mathcal{M} between \mathcal{O}_1 and \mathcal{O}_2 into the bridge rules \mathfrak{B}_{12} between T-Boxes \mathcal{T}_1 and \mathcal{T}_2 in the following way:

- $\langle e, e', \sqsubseteq, n \rangle \mapsto 1:e \xrightarrow{\sqsubseteq} 2:e'$
- $\langle e, e', \supseteq, n \rangle \mapsto 1:e \xrightarrow{\supseteq} 2:e'$
- $\langle e, e', \equiv, n \rangle \mapsto 1:e \xrightarrow{\equiv} 2:e'$

The first element of the semantics of DDL is a local interpretation \mathcal{I}_i for each T-Box \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 T-Boxes $\langle \mathcal{T}_i, \mathcal{T}_j \rangle_{i \neq j}$ elements of the interpretation domains $\Delta^{\mathcal{I}_i}$ and $\Delta^{\mathcal{I}_j}$. $r_{ij}(x)$ is used to denote $\{y \in \Delta^{\mathcal{I}_j} \mid (x, y) \in r_{ij}\}$ 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 a distributed interpretation. The distributed interpretation \mathfrak{I} satisfies a distributed T-Box $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle$ 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 \mathfrak{B}_{ij}
- $r_{ij}(C^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_j}$ for all $i:C \xrightarrow{\supseteq} j:D$ in \mathfrak{B}_{ij}
- $r_{ij}(C^{\mathcal{I}_i}) = D^{\mathcal{I}_j}$ for all $i:C \xrightarrow{\equiv} j:D$ in \mathfrak{B}_{ij}

The key property of DDL bridge rules lays in their ability to propagate knowledge across connected ontologies and to possibly induce changes in subsumption hierarchies in the ontologies. Formally, the ability to derive new subsumption relations across ontologies can be captured by the following propagation rule:

$$\frac{i:A \xrightarrow{\supseteq} j:G, \quad i:B_k \xrightarrow{\sqsubseteq} j:H_k (1 \leq k \leq n), \quad i:A \sqsubseteq \bigsqcup_{k=1}^n B}{j:G \sqsubseteq \bigsqcup_{k=1}^n H_k} \quad (4)$$

Rule (4) completely describes reasoning in DDL that goes beyond well known methods for reasoning in description logics. To be more specific, by adding rule (4) as an additional inference rule to existing tableaux reasoning methods one gets a correct and complete method for reasoning in DDL. A corresponding result using a fixpoint operator is given in (Serafini, Borgida, & Tamilin 2005).

Based on the semantics of DDL, we can now introduce the notion of consistency and entailment of bridge rules and consequently correspondences.¹ The bridge rules of a distributed T-Box \mathfrak{T} can be defined as inconsistent with respect to a local satisfiable concept if the additional constraints induced have the (unintended) effect of making the concept distributed unsatisfiable. If such an effect does not occur the set of bridge rules is consistent with respect to this concept.

Definition 6 (Consistency) *Given a distributed T-Box $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle$, \mathfrak{B}_{ij} is consistent with respect to $j:C$ iff $\mathcal{T}_j \not\models C \sqsubseteq \perp \rightarrow \mathfrak{T} \not\models j:C \sqsubseteq \perp$. 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 .*

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

Further, a bridge rule b can be entailed 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} . The following definition formally introduces the corresponding notion of entailment in general.

Definition 7 (Entailment) *Given a distributed T-Box \mathfrak{T} , a bridge rule b is entailed by \mathfrak{T} iff every model \mathfrak{I} of \mathfrak{T} satisfies b .*

In (Stuckenschmidt, Wache, & Serafini 2006) the authors have described sound and complete algorithms for deciding consistency and entailment of bridge rules. We have implemented these algorithms as an extension of the DRAGO system (Serafini & Tamilin 2005) which we used in the experiments described later.

The extension of the evaluation function can be implemented in a straight forward way by applying the algorithms for checking consistency and entailment. Remember that, given an evaluation function e for a set of bridge

¹For the sake of simplicity we apply the definition of the evaluation function (definition 3) as well as all related definitions in the following to bridge rules. Since there exists a one-to-one assignment between correspondences and bridge rules, a statement about a bridge rule can be understood as the accordant statement about a correspondence.

rules \mathfrak{B}_{ij} between \mathcal{T}_i and \mathcal{T}_j , \mathfrak{B}_{ij} is divided in three complementary subsets $e(\mathfrak{B}_{ij}, \text{correct})$, $e(\mathfrak{B}_{ij}, \text{incorrect})$ and $e(\mathfrak{B}_{ij}, \text{unknown})$. Since all bridge rules in $e(\mathfrak{B}_{ij}, \text{correct})$ are accepted, we can use this information to derive that certain bridge rules in $e(\mathfrak{B}_{ij}, \text{unknown})$ have also implicitly been evaluated, even though the evaluator might not be aware of this. On the one hand, we know that a bridge rule b has to be evaluated as *correct* if b can be entailed by $e(\mathfrak{B}_{ij}, \text{correct})$. On the other hand, we can conclude that each bridge rule b has to be evaluated as *incorrect* if $e(\mathfrak{B}_{ij}, \text{correct}) \cup \{b\}$ is inconsistent.

Algorithm 1

```

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

```

Algorithm 1 is a direct implementation of the above strategy. It takes as input a distributed T-Box $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle$, 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. The algorithm deals with the general case where we might have more than just two ontologies and a complex structure of mappings between those ontologies. In such a situation we might be interested to revise a mapping \mathfrak{B}_{kl} for an additional T-Box \mathcal{T}_l that has just been linked to \mathfrak{T} via \mathfrak{B}_{kl} . Therefore, it is important to take into account the whole structure as well as changes in subsumption hierarchies induced by other mappings that might already have been revised.

We solved the general problem of extending an evaluation in a complex scenario in the following way. First we create a copy e' of e and construct the collection of bridge rules $\{\mathfrak{B}_{-kl}\}$ that consists of all bridge rules except the bridge rules in \mathfrak{B}_{kl} . Then we iterate over the set of bridge rules that have not yet been evaluated as *correct* or *incorrect* and apply the strategy described above to extend the evaluation function e . For the case of entailment we have to reason in a distributed terminology where the set of mappings is restricted to be $\{\mathfrak{B}_{-kl}\} \cup \{e(\mathfrak{B}_{kl}, \text{correct})\}$, while for the case of consistency we also have to add the current bridge rule. Though this approach requires reasoning in a modified distributed T-Box, all modifications are related to the mapping attached to \mathcal{T}_l . This means that the algorithm can be executed locally on the DRAGO reasoning peer hosting \mathcal{T}_l . 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.

Let us revisit the small example stated in an informal way

in the introduction section to illustrate the capabilities of extending an evaluation function.

Example 1 Consider 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:

- (b₁) 1: Document $\xrightarrow{=}$ 2: Document
- (b₂) 1: Abstract $\xrightarrow{=}$ 2: Abstract
- (b₃) 1: Document $\xrightarrow{\supseteq}$ 2: Review

Suppose now that a domain expert for knowledge management evaluates \mathfrak{B}_{12} starting with bridge rule b_1 . He accepts this correspondence and thus we have $e(\mathfrak{B}_{12}, \text{correct}) = \{b_1\}$ and $e(\mathfrak{B}_{12}, \text{unknown}) = \{b_2, b_3\}$. Given the following axioms for \mathcal{T}_1 and \mathcal{T}_2

- $\mathcal{T}_1 \models \text{Document} \sqsupseteq \text{Abstract}$
- $\mathcal{T}_2 \models \text{Document} \sqsubseteq \neg \text{Abstract}$
- $\mathcal{T}_2 \models \text{Document} \sqsupseteq \text{Review}$

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

- $e'(\mathfrak{B}_{12}, \text{correct}) = \{b_1, b_3\}$
- $e'(\mathfrak{B}_{12}, \text{incorrect}) = \{b_2\}$
- $e'(\mathfrak{B}_{12}, \text{unknown}) = \emptyset$

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 large number of evaluation decisions that can be skipped. In the above example for one evaluation decision we gained two further decisions without (direct) manual intervention. The effort of manual intervention can thus be significantly decreased. On the other hand, applying the extension algorithm might sometimes result in non-trivial extensions, in particular where a manual evaluation might result in erroneous decisions. The incorrectness of bridge rule b_2 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 by the logical reasoning implemented in algorithm 1.

Experiments

In our experiments, we focused on the reduction of manual effort of a mapping revision conducted by a domain expert. In the following we measure the manual effort of a revision process in number of evaluation decisions necessary to end up with a completely evaluated set of bridge rules. An evaluation decision is defined to be the specification of a direct successor e' of the previous evaluation function e . As we already argued, the effort for evaluating a set of bridge rules \mathfrak{B} without support will thus be $|\mathfrak{B}|$. Given an evaluation function e for \mathfrak{B} , from definition 5 as well as from the implementation of an extension function (algorithm 1) it follows that computing the extension of e will sometimes result in

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

Table 1: Experimental results for supporting manual evaluation. The first row in each cell compares the size of the mapping to the revised mapping. The second and third row present the fraction of bridge rules to be evaluated manually for different orders. In parentheses you find the number of bridge rules entailed and the number of bridge rules rejected due to inconsistency.

a successor evaluation. Whenever this happens at least one evaluation decision has been computed by logical reasoning and the effort of the expert has been decreased.

We selected four ontologies from the OntoFarm Dataset (Svab *et al.* 2005), referred to as CMT, CRS, PCS and CONFTOOL, and automatically generated mappings between all pairs of ontologies by applying the matching system CtxMatch (Bouquet, Serafini, & Zanobini 2004). In contrast to the majority of existing systems, limited to the discovery of “ \equiv ” correspondences, CtxMatch is additionally capable of finding “ \sqsubset ” and “ \sqsupset ” 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 $\langle \mathcal{O}_i, \mathcal{O}_j \rangle$ with $\mathcal{O}_i \neq \mathcal{O}_j \in \{\text{CMT}, \text{CRS}, \text{PCS}, \text{CONFTOOL}\}$ we built the distributed terminology $\mathcal{T} = \langle \{\mathcal{T}_i, \mathcal{T}_j\}, \{\mathcal{B}_{ij}\} \rangle$ where we interpreted the mapping generated by the CtxMatch matching system as \mathcal{B}_{ij} and the T-Boxes of the ontologies as \mathcal{T}_i and \mathcal{T}_j . Then we proceeded as follows:

1. Init a counter $m \leftarrow 0$ of manual evaluation decisions.
2. Evaluate the first unevaluated bridge rule $b \in \mathcal{B}_{ij}$ and set $m \leftarrow m + 1$.
3. Recompute $e \leftarrow \text{EXTENDEVALUATION}(\mathcal{T}, e, i, j)$.
4. If $e(\mathcal{B}_{ij}, \text{unknown}) \neq \emptyset$ 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, $\frac{m}{|\mathcal{B}_{ij}|}$ 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 as the number of bridge rules that have been evaluated as *incorrect* by checking consistency.

In a first series of experiments we ordered the bridge rules in a random way.² The results for these experiments are

²More precisely, to make the results reproducible we ordered

presented in the second row of each cell in table 1. 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.

Even though these results show the benefit of our approach, there is still room for improvement by presenting the bridge rules in a proper order to the domain expert. The following example describes the effects of different orderings.

Example 2 Consider the set of bridge rules $\mathcal{B}_{12} = \{b_1, b_2, b_3\}$ from example 1. Taking into account the semantics of the ontologies that are aligned via \mathcal{B}_{12} , the following applies:

- b_1 and b_3 are correct and b_2 is incorrect,
- $\{b_1\}$ entails b_3 ,
- and $\{b_1, b_2\}$ is inconsistent.

If we first present b_1 to the domain expert and extend the resulting evaluation function we end up with a complete evaluation function e with $e(\mathcal{B}, \text{unknown}) = \emptyset$. This has already been shown in example 1. Compare this to the results that we achieve if we present the bridge rules in the sequence $\langle b_3, b_2, b_1 \rangle$. In this case the evaluation cannot be extended at all. This applies for each step of the revision process based on this ordering.

Example 2 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 8. 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

the bridge rules lexicographically with respect to the names of concepts matched by the bridge rule.

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 bridge rules.

Definition 8 (Potential impact of a bridge rule) *The potential impact of a bridge rule from \mathcal{T}_1 to \mathcal{T}_2 denoted as $imp(\mathcal{T}_1, \mathcal{T}_2, 1: C \xrightarrow{R} 2: D)$ is defined as*

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

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 1 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.

Discussion

In a recent study (Falconer & Storey 2007) Falconer and Storey discuss requirements of systems used for mapping creation and verification with respect to the user point of view. Amongst others, they identify conflict resolution and inconsistency detection as an important requirement for such tools. The corresponding functionality of existing systems, however, is restricted to structural consistency criteria. In this paper, we argued for the need of providing reasoning support for manual mapping revision. We pointed out that the inherent complexity and size of the problem makes the revision process an error-prone and time-consuming task. Therefore, we proposed a reasoning method to extend a partial evaluation of a mapping based on logical reasoning. We argued that this kind of reasoning can be used to both detect incorrect correspondences, that are hard to find for a human expert, and decrease human effort in terms of correspondences that have to be evaluated. In our experiments, we showed that the manual effort can be reduced to a significant degree by applying our approach.

Nevertheless, there are a number of open problems that need to be addressed in future work. In particular, the detection of inconsistencies in a mapping relies on the presence of disjointness axioms. In practice these axioms are often missing. Notice that the low number of incorrect correspondences detected by consistency checking reported in the experimental section is based on this fact. There are several

ways to deal with this problem. One is to work with the assumption that sibling-concepts are always disjoint, adding the corresponding axioms to the ontologies. This has already successfully been done in the context of revising ontologies (Schlobach 2005). In current work we also explore the option of automatically creating the required disjointness statements using machine learning techniques.

Another potential problem is the complexity of the reasoning problem involved. Recently in (Meilicke & Stuckenschmidt 2007), Meilicke and Stuckenschmidt have explored efficient approximations of reasoning services for checking mapping consistency. Such approximations only require to classify the ontologies once and then use correct but incomplete reasoning heuristics for checking mapping consistency. The practical evaluation showed that only in a very few examples the approximate method failed to detect all inconsistencies. The similar approximation approach could be used to check the entailment of correspondences to scale the mapping revision process proposed in this paper.

References

- Bouquet, P.; Serafini, L.; and Zanobini, S. 2004. Peer-to-peer semantic coordination. *Journal of Web Semantics* 2(1):81–97.
- Euzenat, J., and Shvaiko, P. 2007. *Ontology Matching*. Springer.
- Falconer, S. M., and Storey, M.-A. 2007. A cognitive support framework for ontology mapping. In *Proceedings of the ISWC 2007*, 114–127. Springer.
- Meilicke, C., and Stuckenschmidt, H. 2007. Applying logical constraints to ontology matching. In *Proceedings of the 30th German Conference on Artificial Intelligence (KI-07)*.
- Meilicke, C.; Stuckenschmidt, H.; and Tamin, A. 2007. Repairing ontology mappings. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*.
- Schlobach, S. 2005. Debugging and semantic clarification by pinpointing. In *Proceedings of ESWC 2005*.
- Serafini, L., and Tamin, A. 2005. DRAGO: Distributed reasoning architecture for the semantic web. In *Proceedings of the Second European Semantic Web Conference (ESWC'05)*.
- Serafini, L.; Borgida, A.; and Tamin, A. 2005. Aspects of distributed and modular ontology reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI-05*.
- Stuckenschmidt, H.; Wache, H.; and Serafini, L. 2006. Reasoning about ontology mappings. In *Proceedings of the ECAI-06 Workshop on Contextual Representation and Reasoning*.
- 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*.