A Conflict-based Operator for Mapping Revision
Theory and Implementation

Guilin Qi[1,2], Qiu Ji[1], Peter Haase[1]

[1]Institute AIFB, University of Karlsruhe, Germany
{gqi,qiji,pha}@aifb.uni-karlsruhe.de
[2] School of Computer Science and Engineering
Southeast University**, Nanjing 210096

Abstract. Ontology matching is one of the key research topics in the field of the Semantic Web. There are many matching systems that generate mappings between different ontologies either automatically or semi-automatically. However, the mappings generated by these systems may be inconsistent with the ontologies. Several approaches have been proposed to deal with the inconsistencies between mappings and ontologies. This problem is often called a mapping revision problem, as the ontologies are assumed to be correct, whereas the mappings are repaired when resolving the inconsistencies. In this paper, we first propose a conflict-based mapping revision operator and show that it can be characterized by two logical postulates adapted from some existing postulates for belief base revision. We then provide an algorithm for iterative mapping revision by using an ontology revision operator and show that this algorithm defines a conflict-based mapping revision operator. Three concrete ontology revision operators are given to instantiate the iterative algorithm, which result in three different mapping revision algorithms. We implement these algorithms and provide some preliminary but interesting evaluation results.

1 Introduction

Next generation semantic applications are employed by a large number of ontologies, some of them constantly evolving. As the complexity of semantic applications increases, more and more knowledge is embedded in ontologies, typically drawn from a wide variety of sources. This new generation of applications thus likely relies on a set of distributed ontologies, typically connected by mappings. One of the major challenges in managing these distributed and dynamic ontologies is to handle potential inconsistencies introduced by integrating multiple distributed ontologies.

For inconsistency handling in single, centralized ontologies, several approaches are known (see the survey in [7]). Recently, there are some works done on handling inconsistency in distributed ontologies connected by mappings, where a mapping between two ontologies is a set of correspondences between entities

** New affiliation since September 2009.

in the ontologies. In a distributed system consisting of two ontologies and a mapping between them, correspondences in the mapping can have different interpretations. For example, in Distributed Description Logics (DDL) [3], a correspondence in a mapping is translated into two *bridge rules* that describe the "flow of information" from one ontology to another one. In [13], the authors deal with the problem of mapping revision in DDL by removing some bridge rules which are responsible for the inconsistency. The idea of their approach is similar to that of the approaches for debugging and repairing terminologies in a single ontology. Mappings can also be interpreted as sets of axioms in a description logic. A heuristic method for mapping revision is given in [12]. However, this method can only deal with inconsistency caused by disjointness axioms which state that two concepts are disjoint. Later on, Meilicke et al. proposed another algorithm to resolve the inconsistent mappings in [15]. The idea of their algorithm is similar to the linear base revision operator given in [16]. However, both methods given in [12] and [15] lack a rationality analysis w.r.t. logical properties.

In this paper, we first propose a conflict-based mapping revision operator based on the notion of a "conflict set", which is a subset of the mapping that is in conflict with ontologies in a distributed system. We then adapt two postulates from belief revision theory [8] and show that our mapping revision operator can be characterized by them (see Section 3). After that, in Section 4, we provide an iterative algorithm for mapping revision by using a revision operator in description logics and show that this algorithm results in a conflict-based mapping revision operator. We define a revision operator and show that the iterative algorithm based on it produces the same results as the algorithm given in [15]. This specific iterative algorithm has a polynomial time complexity if the satisfiability check of an ontology can be done in polynomial time in the size of the ontology. However, this algorithm may still be inefficient for large ontologies and mappings, because it requires a large number of satisfiability checks. Therefore, we provide an algorithm to implement an alternative revision operator based on the *relevance-based selection function* given in [11] which can be optimized by a module extraction technique given in [22]. Neither of the above proposed revision operators removes minimal number of correspondences to resolve inconsistencies. To better fulfil the principle of minimal change, we consider the revision operator given in [19] which utilizes a heuristics based on a *scoring function* which returns the number of *minimal incoherence-preserving sub-ontologies (MIPS)* that an axiom belongs to. Instantiating our iterative algorithm with this existing revision operator results in a new conflict-based mapping revision operator. Finally, we implement these algorithms and provide evaluation results for comparing their efficiency and effectiveness in Section 5.

Relationship with belief revision This work is related to belief revision which has been widely discussed in the literature [5, 10]. Our conflict-based mapping revision operator is inspired by the internal revision operator given in [8], and the postulate used to characterize our mapping revision operator is adapted from a postulate for internal revision operator given in [8]. The problem of mapping revision is not exactly the same as the problem of belief base revision

because the mapping to be revised is dependent on ontologies in the distributed system and each correspondence in the mapping carries a confidence value which can be used to guide the revision. Our iterative algorithm is inspired by the iterative revision algorithm given in [18] and is tailored to produce a conflict-based revision operator.

2 Preliminaries

We assume that the reader is familiar with Description Logics (DL) and refer to Chapter 2 of the DL handbook [1] for a good introduction. Our method is independent of a specific DL language, and thus can be applied to any DL.

A DL-based ontology (or knowledge base) $O = (\mathcal{T}, \mathcal{R})$ consists of a set $\mathcal{T}$ of concept axioms (TBox) and a set $\mathcal{R}$ of role axioms (RBox). In this paper, we treat O as a set of axioms. Concept axioms (or terminology axioms) have the form $C \sqsubseteq D$, where C and D are (possibly complex) concept descriptions built from a set of concept names and some constructors, and role axioms are expressions of the form $R \sqsubseteq S$, where R and S are (possibly complex) role descriptions built from a set of role names and some constructors.

An interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain set $\triangle^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$, which maps from concepts and roles to subsets of the domain and binary relations on the domain, respectively. Given an interpretation $\mathcal{I}$, we say that $\mathcal{I}$ satisfies a concept axiom $C \sqsubseteq D$ (resp., a role inclusion axiom $R \sqsubseteq S$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (resp., $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$). An interpretation $\mathcal{I}$ is called a *model* of an ontology O, iff it satisfies each axiom in O. A concept C in an ontology O is unsatisfiable if for each model $\mathcal{I}$ of O, $C^{\mathcal{I}} = \emptyset$. An ontology O is incoherent if there exists an unsatisfiable concept in O.

Given two ontologies O_1 and O_2, describing the same or largely overlapping domains of interest, we can define correspondences between their elements.

Definition 1. *[4] Let O_1 and O_2 be two ontologies, Q be a function that defines sets of mappable elements $Q(O_1)$ and $Q(O_2)$. A correspondence is a 4-tuple $\langle e, e', r, \alpha \rangle$ such that $e \in Q(O_1)$ and $e' \in Q(O_2)$, r is a semantic relation, and α is a confidence value from a suitable structure $\langle D, \leq \rangle$, such as a lattice. A mapping $\mathcal{M}$ is a set of correspondences.*

In Definition 1, there is no restriction on function Q, semantic relation r and domain D. In the mapping revision scenario, we often consider correspondences between concepts and restrict r to be one of the semantic relations from the set $\{\equiv, \sqsubseteq, \sqsupseteq\}$, and let $D = [0, 1]$. A mapping is a set of correspondences whose elements are mappable. The following definition is adapted from the definition of a distributed system given in [23].

Definition 2. *A distributed system is a triple $\mathcal{D} = \langle O_1, O_2, \mathcal{M} \rangle$, where O_1 and O_2 are ontologies and $\mathcal{M}$ is a mapping between them. We call O_1 the source ontology and O_2 the target ontology.*

Example 1. Take the two ontologies CRS and EKAW in the domain of conference management systems as an example. They contain the following axioms:

$\mathsf{crs:article} \sqsubseteq \mathsf{crs:document}$, $\mathsf{crs:program} \sqsubseteq \neg\mathsf{crs:document}$,
$\mathsf{ekaw:Paper} \sqsubseteq \mathsf{ekaw:Document}$, $\mathsf{ekaw:Workshop\_Paper} \sqsubseteq \mathsf{ekaw:Paper}$
$\mathsf{ekaw:Conference\_Paper} \sqsubseteq \mathsf{ekaw:Paper}$, $\mathsf{ekaw:PC\_Member} \sqsubseteq \mathsf{ekaw:Possible\_Reviewer}$,

The correspondences in the mapping $\mathcal{M}$ between O_1 and O_2 which is obtained by the ontology matching system HMatch are listed as follows:

$\mathsf{m}_1 : \langle \mathsf{crs:article}, \mathsf{ekaw:Conference\_Paper}, \sqsubseteq, 0.65 \rangle$
$\mathsf{m}_2 : \langle \mathsf{ekaw:Workshop\_Paper}, \mathsf{crs:article}, \sqsubseteq, 0.65 \rangle$
$\mathsf{m}_3 : \langle \mathsf{ekaw:Document}, \mathsf{crs:program}, \sqsubseteq, 0.80 \rangle$
$\mathsf{m}_4 : \langle \mathsf{crs:program}, \mathsf{ekaw:Document}, \sqsubseteq, 0.80 \rangle$
$\mathsf{m}_5 : \langle \mathsf{crs:document}, \mathsf{ekaw:Document}, \sqsubseteq, 0.93 \rangle$

Definition 3. *[13] Let $\mathcal{D} = \langle O_1, O_2, \mathcal{M} \rangle$ be a distributed system. The* union *$O_1 \cup_{\mathcal{M}} O_2$ of O_1 and O_2 connected by $\mathcal{M}$ is defined as $O_1 \cup_{\mathcal{M}} O_2 = O_1 \cup O_2 \cup \{t(m) : m \in \mathcal{M}\}$ with t being a translation function that converts a correspondence into an axiom in the following way: $t(\langle C, C', r, \alpha \rangle) = CrC'$.*

That is, we first translate all the correspondences in the mapping $\mathcal{M}$ into DL axioms, then the union of the two ontologies connected by the mapping is the set-union of the two ontologies and the translated axioms. Given $\mathcal{D} = \langle O_1, O_2, \mathcal{M} \rangle$, we use $Union(\mathcal{D})$ to denote $O_1 \cup_{\mathcal{M}} O_2$. Take a correspondence in Example 1 as an example, we have $\mathsf{t}(\langle \mathsf{crs:article}, \mathsf{ekaw:Conference\_Paper}, \sqsubseteq, 0.65 \rangle) = \mathsf{crs:article} \sqsubseteq \mathsf{ekaw:Conference\_Paper}$.

Definition 4. *[12] Given a mapping $\mathcal{M}$ between two ontologies O_1 and O_2, $\mathcal{M}$ is consistent with O_1 and O_2 iff there exists no concept C in O_i with $i \in \{1, 2\}$ such that C is satisfiable in O_i but unsatisfiable in $O_1 \cup_{\mathcal{M}} O_2$. Otherwise, $\mathcal{M}$ is inconsistent. A distributed system $\mathcal{D} = \langle O_1, O_2, \mathcal{M} \rangle$ is inconsistent if $\mathcal{M}$ is inconsistent with O_1 and O_2.*

An inconsistent mapping is a mapping such that there is a concept that is satisfiable in a mapped ontology but unsatisfiable in the union of the two ontologies together with the mapping. In Example 1, since ekaw:Workshop_Paper is satisfiable in both O_1 and O_2 but unsatisfiable in $O_1 \cup_{\mathcal{M}} O_2$, $\mathcal{M}$ is inconsistent. Note that $O_1 \cup O_2$ must be coherent if both O_1 and O_2 are coherent because they use different name spaces.

Definition 5. *A mapping revision operator $\circ$ is a function $\circ\langle O_1, O_2, \mathcal{M} \rangle = \langle O_1, O_2, \mathcal{M}' \rangle$ such that $\mathcal{M}' \subseteq \mathcal{M}$, where O_1 and O_2 are two ontologies and $\mathcal{M}$ is a mapping between them.*

Our definition of a mapping revision operator is similar to the definition of a revision function given in [14]. When repairing the mapping in a distributed system, we assume that ontologies are more reliable than the mapping and therefore only remove correspondences in the mapping to restore consistency. This

makes the problem of mapping repair akin to the problem of belief revision. Thus we call the problem of repairing mappings *mapping revision*. However, this definition is very general and allows mapping revision operators that result in unintuitive results. That is, we can define two naive revision operators $\circ_{Full}\langle O_1, O_2, \mathcal{M}\rangle = \langle O_1, O_2, \emptyset\rangle$ and $\circ_{Null}\langle O_1, O_2, \mathcal{M}\rangle = \langle O_1, O_2, \mathcal{M}\rangle$. In belief revision, the rationality of a revision operator is often evaluated by logical postulates. In this work, we will define a mapping revision operator and show that it can be characterized by an important logical postulate.

3 A Conflict-based Mapping Revision Operator

In this section, we propose a method for mapping revision based on the idea of kernel contractions defined by Hansson in [9]. We adapt the notion of a minimal conflict set of a distributed system given in [13] as follows.

Definition 6. *Let $\langle O_1, O_2, \mathcal{M}\rangle$ be a distributed system. A subset $\mathcal{C}$ of $\mathcal{M}$ is a* conflict set *for a concept A in O_i ($i = 1, 2$) if A is satisfiable in O_i but unsatisfiable in $O_1 \cup_{\mathcal{C}} O_2$. $\mathcal{C}$ is a* minimal conflict set *(MCS) for A in O_i if $\mathcal{C}$ is a conflict set for A and there exists no $\mathcal{C}' \subset \mathcal{C}$ which is also a conflict set for A in O_i.*

A minimal conflict set for a concept in one of the ontologies is a minimal subset of the mapping that, together with the ontologies, is responsible for the unsatisfiability of the concept in the distributed system. It is similar to the notion of a kernel in [9]. Note that if O_i ($i = 1, 2$) is incoherent, then it is meaningless to define the notion of a MCS for an unsatisfiable concept. We use $MCS_{O_1,O_2}(\mathcal{M})$ to denote the set of all the minimal conflict sets for all unsatisfiable concepts in $O_1 \cup_{\mathcal{C}} O_2$. It corresponds to the notion of a *kernel set* in [9]. In Example 1, $MCS_{CRS,EKAW}(\mathcal{M}) = \{\{t(m_1), t(m_3)\}, \{t(m_2), t(m_3)\}, \{t(m_3), t(m_5)\}, \{t(m_1), t(m_2), t(m_3)\}, \{t(m_2), t(m_3), t(m_5)\}\}$.

Hansson's kernel contraction removes formulas in a knowledge base through an *incision function*, which is a function that selects formulas to be discarded. However, we cannot apply the notion of an incision function to mapping revision directly because the mapping to be revised is dependent on the ontologies in the distributed system. Therefore, the problem of mapping revision is not exactly the same as the problem of belief revision where the two knowledge bases may come from different sources. Furthermore, each correspondence in the mapping carries a confidence value which can be used to guide the revision. We use $\mathbf{D}$ and $\mathbf{M}$ to denote the set of all the distributed systems and the set of all the subsets of mappings in all distributed systems.

Definition 7. *An* incision function *σ: $\mathbf{D} \rightarrow \mathbf{M}$ is a function such that for any distributed system $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$, we have*

(i) $\sigma(\mathcal{D}) \subseteq \bigcup(MCS_{O_1,O_2}(\mathcal{M}))$;
(ii) if $\mathcal{C} \neq \emptyset$ *and* $\mathcal{C} \in MCS_{O_1,O_2}(\mathcal{M})$, *then* $\mathcal{C} \cap \sigma(\mathcal{D}) \neq \emptyset$;

(iii) if $m = \langle C, C', r, \alpha\rangle \in \sigma(\mathcal{D})$*, then there exists* $\mathcal{C} \in MCS_{O_1,O_2}(\mathcal{M})$ *such that* $m \in \mathcal{C}$ *and* $\alpha = min\{\alpha_i : \langle C_i, C'_i, r_i, \alpha_i\rangle \in \mathcal{C}\}$.

The first two conditions say that an incision function selects from each kernel set at least one element. The third condition says that if a correspondence is selected by an incision function, then there must exist a MCS $\mathcal{C}$ such that its confidence value is the minimal confidence value of correspondences in $\mathcal{C}$. Going back to Example 1, the incision function σ may select m_1, m_2 and m_3 to resolve inconsistency.

We define our mapping revision operator based on an incision function.

Definition 8. *A mapping revision operator* $\circ$ *is called a* conflict-based mapping revision operator *if there exists an incision function* σ *such that:*

$$\circ\langle O_1, O_2, \mathcal{M}\rangle = \langle O_1, O_2, \mathcal{M} \setminus \sigma(MCS_{O_1,O_2}(\mathcal{M}))\rangle.$$

That is, we remove those correspondences in $\mathcal{M}$ that are selected by the incision function to restore consistency. We provide the representation theorem for conflict-based mapping revision. Before that, we need to define the notion of an inconsistency degree of a distributed system for a concept. Given a distributed system $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$, a concept A in O_i $(i = 1, 2)$ is unsatisfiable in $\mathcal{D}$ if A is unsatisfiable in $O_1 \cup_{\mathcal{M}} O_2$.

Definition 9. *Given* $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$*, the* β*-cut (resp. strict* β*-cut) set of* $\mathcal{D}$*, denoted as* $\mathcal{D}_{\geq\beta}$ *(resp.* $\mathcal{D}_{>\beta}$*), is defined as* $\mathcal{D}_{\geq\beta} = \langle O_1, O_2, \{\langle C, C', r, \alpha\rangle \in \mathcal{M} : \alpha \geq \beta\}\rangle$ *(resp.* $\mathcal{D}_{>\beta} = \langle O_1, O_2, \{\langle C, C', r, \alpha\rangle \in \mathcal{M} : \alpha > \beta\}\rangle$*).*

The β-cut set of $\mathcal{D}$ is a distributed system consisting of O_1, O_2 and correspondences in the mapping whose confidence values are greater than or equal to β. It is adapted from the notion of cut set in possibilistic DL in [20]. In Example 1, $\mathcal{D}_{>0.65} = \langle O_1, O_2, \{t(m_3), t(m_4), t(m_5)\}\rangle$.

Definition 10. *Given* $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$*, the inconsistency degree of* $\mathcal{D}$ *for a concept* A *in* O_i $(i = 1, 2)$*, denoted by* $Inc(\mathcal{D})_A$*, is defined as* $Inc(\mathcal{D})_A = max\{\alpha : A \text{ is unsatisfiable in } \mathcal{D}_{\geq\alpha}\}$*. The inconsistency degree of* $\mathcal{D}$*, denoted as* $Inc(\mathcal{D})$*, is defined as* $Inc(\mathcal{D}) = max\{\alpha : \text{there exists an unsatisfiable concept in } \mathcal{D}_{\geq\alpha}\}$.

It is easy to check that $Inc(\mathcal{D}) = max\{\alpha : \mathcal{D}_{\geq\alpha} \text{ is inconsistent}\}$. In Example 1, $\mathcal{D}_{\geq 0.93}$ is consistent but $\mathcal{D}_{\geq 0.8}$ is inconsistent since ekaw:Workshop_Paper is unsatisfiable. Thus, $Inc(\mathcal{D}) = 0.8$.

We give a postulate for mapping revision by generalizing the postulate (Relevance) for the internal partial meet revision operator given in [8]. It says that if a correspondence is removed from the mapping after revision, then it must be in a conflict set of the mapping for a concept and the confidence degree attached to it is minimal among all the confidence degrees in the conflict set.

Postulate (Relevance) Suppose $\circ\langle O_1, O_2, \mathcal{M}\rangle = \langle O_1, O_2, \mathcal{M}'\rangle$, if $m = \langle C, C', r, \alpha\rangle \in \mathcal{M}$ and $m \notin \mathcal{M}'$, then there exists a concept A in O_i $(i = 1, 2)$ and a

Algorithm 1: An iterative algorithm for mapping revision

Data: A distributed system $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$ and a revision operator $\star$
Result: A repaired distributed system $\mathcal{D}_\star = \langle O_1, O_2, \mathcal{M}_\star\rangle$

```
1  begin
2      if either O1 or O2 is incoherent then
3          return D
4      Rearrange the weights in M such that β1>β2>...>βl > 0;
5      Si := {t(⟨C, C′, r, α⟩) : ⟨C, C′, r, α⟩∈M, α = βi}, i = 1, ..., l;
6      while M in D is inconsistent do
7          if βk = Inc(D) then
8              St := Sk \ (Sk ⋆ (Union(D)>βk));
9              M := M \ {⟨C, C′, r, α⟩ : t(⟨C, C′, r, α⟩) ∈ St, α = βk};
10     return D
11 end
```

subset $\mathcal{S}$ of $\mathcal{M}$ such that A is satisfiable in $\langle O_1, O_2, \mathcal{S}\rangle$ but is unsatisfiable in $\langle O_1, O_2, \mathcal{S} \cup \{m\}\rangle$ and $Inc(\langle O_1, O_2, \mathcal{S} \cup \{m\}\rangle)_A = \alpha$.

Relevance is an important postulate for minimal change. However, it does not constrain the number of correspondences to be removed. Therefore, it does not entail minimal change.

We also need another postulate called *Consistency*.

Postulate (Consistency) For any $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$ where O_i are coherent, $\circ\langle O_1, O_2, \mathcal{M}\rangle$ is a consistent distributed system.

The following theorem shows that our conflict-based mapping revision operator can be characterized by the postulates (Relevance) and (Consistency).

Theorem 1. *The operator $\circ$ is a conflict-based mapping revision operator if and only if it satisfies (Relevance) and (Consistency).*

To show the *if* direction of the theorem, we can construct $\sigma(\mathcal{D}) = \mathcal{M} \setminus \mathcal{M}'$ for $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$ and $\circ(\mathcal{D}) = \langle O_1, O_2, \mathcal{M}'\rangle$, then show that σ is an incision function. Unlike revision operators given in [8], our conflict-based mapping revision operator is characterized by only two postulates. This is because the definition of a conflict already gives some constraints on how we can repair a mapping. According to Definition 5, ontologies in the distributed systems are not changed and revised mapping must be a subset of the original one. These two conditions correspond to (Success) and (Inclusion) for revision operators given in [8].

4 An Algorithm for Mapping Revision

In this section, we give an algorithm for mapping revision based on an ontology revision operator and then present some concrete ontology revision operators.

4.1 Algorithm

We describe the idea of our algorithm (Algorithm 1) as follows. Given a distributed system $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$, if either O_1 or O_2 is incoherent, then we

take $\mathcal{D}$ as the result of revision. That is, no change is needed. Suppose $\mathcal{M} = \{\langle C_i, C_i', r_i, \alpha_i\rangle : i = 1, ..., n\}$ where n is the number of correspondences in $\mathcal{M}$. Let us rearrange the weights of axioms (i.e., α_i) in $\mathcal{M}$ such that $\beta_1 > \beta_2 > ... > \beta_l > 0$, where β_i $(i = 1, ..., l)$ are all the distinct weights appearing in $\mathcal{M}$. For each $i \in \{1, ..., l\}$, S_i consists of translated axioms of correspondences in $\mathcal{M}$ which have the confidence value β_i. Suppose $Inc(\mathcal{D}) = \beta_k$. We revise S_k by $Union(\mathcal{D}_{>\beta_k})$. Suppose S_t is the set of axioms in S_k that are removed after revision of S_k by $Union(\mathcal{D}_{>\beta_k})$ using the operator $\star$. We then remove the correspondences in $\mathcal{M}$ that have confidence values β_k and are mapped to axioms in S_t by the translation function t. We iterate the revision process until the mapping becomes consistent.

In Algorithm 1, we need to compute the inconsistency degree of a distributed system. This can be easily done by adapting the algorithm for computing the inconsistency degree in [20] so we do not bother to provide it here.

We have not specified a revision operator in Algorithm 1. However, we require that the revision operator $\star$ used in the algorithm satisfy the following properties which are similar to the postulates *Inclusion*, *Success* and *Core-retainment* for kernel revision operator given in [9]:

- Inclusion: $O \star O' \subseteq O \cup O'$;
- Success: $O' \subseteq O \star O'$;
- Core-retainment: if $\phi \in O$ and $\phi \notin O \star O'$, then there exist a concept A in $O \cup O'$ and a subset O_s of O, such that A is satisfiable in $O_s \cup O'$ but is unsatisfiable in $O_s \cup O' \cup \{\phi\}$.

It is clear that Algorithm 1 generates a mapping revision operator. We show that this operator is a conflict-based mapping revision operator.

Theorem 2. *Suppose $\star$ satisfies Inclusion, Success and Core-retainment, and $\circ$ is a mapping revision operator such that, for any distributed system $\mathcal{D}$, $\circ(\mathcal{D})$ is the result of Algorithm 1 with $\star$ as an input parameter, then $\circ$ is a conflict-based mapping revision operator.*

4.2 Concrete revision operators

We first give a simple revision operator which is adapted from the linear base revision operator given in [16]. By SORT we denote a procedure that for each ontology $O = \{\phi_1, ..., \phi_n\}$, randomly ranks its elements as an ordered sequence $(\phi_1, ..., \phi_n)$. Let O and O' be two ontologies, and let $\text{SORT}(O) = \{\phi_1, ..., \phi_n\}$, the random linear base revision operator, denoted as $\circ_{linear}$, is defined inductively as follows $O \circ_{linear} O' = O' \cup S_1 \cup ... \cup S_n$, where S_i is defined by $S_i = \{\phi_i\}$ if $\{\phi_i\} \cup O' \cup \bigcup_{j=1}^{i-1} S_j$ is coherent, $\emptyset$ otherwise, for $i \geq 1$. It is easy to check that this revision operator satisfies conditions Inclusion, Success and Core-retainment. We show that the algorithm given in [15] is a special case of our iterative algorithm where the operator $\circ_{linear}$ is chosen.

Proposition 1. *For any distributed system $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$ where O_1 and O_2 are coherent, suppose $\mathcal{D}_{\circ_{linear}}$ is the result of revision by Algorithm 1, then $\mathcal{D}_{\circ_{linear}}$ can be obtained by the algorithm given in [15] as well.*

REL_REVISION(O, O′) **Input**: Two ontologies O and O' **Output**: A revised ontology $O \star O'$	CONF(C, O, O′) **Input**: Two ontologies O and O', and an unsatisfiable concept C of $O \cup O'$ **Output**: A hitting set hs in O for C w.r.t. O'

(1) Global : $\mathcal{J} \leftarrow \emptyset$;
(2) HS $\leftarrow \emptyset$;
(3) for(C $\in$ AllUnsatConcepts(O $\cup$ O′)){
(4) k $\leftarrow$ 1;
(5) $O_t \leftarrow$ hs $\leftarrow \emptyset$;
(6) while(s_k(O $\cup$ O′, C) $\neq \emptyset$){
(7) $O_t \leftarrow O_t \cup s_k$(O $\cup$ O′, C);
(8) if(hs $\neq \emptyset$){
(9) if((O $\setminus$ hs) $\cup$ O′ $\not\models$ C $\sqsubseteq \bot$)
(10) break;
(11) hs $\leftarrow$ CONF(C, $O_t \cap$ (O $\setminus$ hs), $O_t \cap$ O′);
(12) HS $\leftarrow$ HS $\cup$ hs;
(13) }else if($O_t \models$ C $\sqsubseteq \bot$){
(14) hs $\leftarrow$ CONF(C, $O_t \cap$ O, $O_t \cap$ O′);
(15) HS $\leftarrow$ HS $\cup$ hs;
(16) }
(17) k $\leftarrow$ k + 1;
(18) } (end_while)
(19) } (end_for)
(20) return (O $\setminus$ HS) $\cup$ O′;
	(1) hs $\leftarrow \emptyset$;
	(2) while((O $\setminus$ hs) $\cup$ O′ $\models$ C $\sqsubseteq \bot$){
	(3) J $\leftarrow$ SINGLE_CONFLICT(C, O $\setminus$ hs, O′);
	(4) $\mathcal{J} \leftarrow \mathcal{J} \cup$ {J}
	(5) hs = hs $\cup$ {ϕ} for some $\phi \in$ J;
	(6) }
	(7) return hs;

Table 1. Relevance-based mapping revision algorithm.

As shown in [15], their algorithm only needs at most n satisfiability check, where n is the number of correspondences. Therefore, our iterative algorithm based on the revision operator $\circ_{linear}$ has a polynomial time complexity if the satisfiability check can be done in polynomial time in the size of union of ontologies and the mapping. However, this algorithm requires to rank correspondences with the same confidence value and there is no good principle to guide this ranking. Furthermore, if the size of the union of ontologies and the mapping is big, then the algorithm may still be inefficient because it will need a large number of satisfiability checks over the union.

In the following, we present an algorithm REL_REVISION (see Table 1) to implement another concrete revision operator based on the relevance-based selection function. The motivation behind the algorithm is that when choosing between two correspondences to remove, we always remove the one which is more relevant to an unsatisfiable concept and thus is more likely to be problematic.

Given two axioms ϕ and ψ, ϕ is *directly relevant* to ψ iff there is an overlap between the signature of ϕ and the signature of ψ, where the signature of an axiom is the set of all concept names, role names and individual names appearing in it. Based on the notion of direct relevance, we can extend it to relevance relation between an axiom and an ontology. An axiom ϕ is relevant to an ontology O iff there exists an axiom ψ in O such that ϕ and ψ are directly relevant. We introduce a selection function defined in [11].

Definition 11. *[11] Let O be an ontology, ϕ be an axiom and k be an integer. The* relevance-based selection function*, written s_{rel}, is defined inductively as follows:*
$s_{rel}(O, \phi, 0) = \emptyset$
$s_{rel}(O, \phi, 1) = \{\psi \in O : \phi$ *is directly relevant to* $\psi\}$

$s_{rel}(O, \phi, k) = \{\psi \in O : \psi$ *is directly relevant to* $s_{rel}(O, \phi, k-1)\}$, *where* $k > 1$.

We call $s_{rel}(O, \phi, k)$ *the* k*-relevant subset of* O *w.r.t.* ϕ. *For convenience, we define* $s_k(O, \phi) = s_{rel}(O, \phi, k) \setminus s_{rel}(O, \phi, k-1)$ *for* $k \geq 1$.

Our algorithm REL_REVISION is based on Reiter's Hitting Set Tree (HST) algorithm [21]. Given a *universal set* U, and a set $K = \{s_1, ..., s_n\}$ of subsets of U which are *conflict sets*, i.e. subsets of the system components responsible for the error, a *hitting set* T for K is a subset of U such that $s_i \cap T \neq \emptyset$ for all $1 \leq i \leq n$. To adapt HST algorithm to deal with revision of ontologies in DLs, we define the notion of a *minimal conflict set* of an ontology O for a concept C w.r.t. another ontology O'. A subset O_s of O is called a minimal conflict set of O for C w.r.t. O', if (1) C is unsatisfiable in $O_s \cup O'$ and (2) for any $O_t \subset O_s$, C is satisfiable in $O_t \cup O'$. A more general definition of a minimal conflict set is given in [2], where it is called a *minimal axiom set*.

In REL_REVISION, we handle unsatisfiable concepts in the union of the mapped ontologies and the ontology translated from the mapping one by one until we resolve the inconsistency. For each unsatisfiable concept to be handled, we first select axioms that are relevant to it iteratively by the relevance-based selection function until the concept is unsatisfiable in these axioms. $s_k(O, C)$ is the abbreviation of $s_k(O, C \sqsubseteq \bot)$. We find a hitting set for the selected sub-ontologies by calling the procedure CONF and update the existing incomplete hitting set HS. We then add to the selected sub-ontologies those axioms that are directly relevant to them and further expand the hitting set tree by calling to procedure CONF. We continue this process until the inconsistency is resolved. The procedure SINGLE_CONFLICT computes a minimal conflict set of O for C w.r.t. O'. This kind of procedure can be found in the literature, such as GETMUPS in [19]. It is possible that some axioms that are involved in a conflict set are not selected by the selection function. Therefore, when $s_k(O \cup O', C) = \emptyset$, we still have $(O \setminus HS) \cup O' \models C \sqsubseteq \bot$, then we set $s_k(O \cup O', C) = (O \cup O') \setminus s_{rel}(O \cup O', C \sqsubseteq \bot, k-1)$. Note that our algorithm may not remove minimal number of correspondences to resolve inconsistency because we only expand one branch of the hitting set tree in a depth-first manner. This is compensated by higher efficiency. Furthermore, although our algorithm does not remove minimal number of correspondences, the removals of correspondences are guided by a relevance-based selection function to improve the quality of removal. It is easy to see that the revision operator obtained by REL_REVISION satisfies conditions Inclusion, Success and Core-retainment.

In REL_REVISION, to resolve an unsatisfiable concept C in $O \cup O'$, we need to compute some minimal conflict sets of O for C w.r.t. O'. The time complexity of REL_REVISION depends on the DL under consideration. In the worst case, i.e., all the minimal conflict sets of all the unsatisfiable concepts are disjoint, our algorithm needs to compute all the minimal conflict sets for all the unsatisfiable concepts, which is a hard task [17]. For instance, the number of all the minimal conflict sets for an unsatisfiable concept is exponential in the worst case for lightweight ontology language EL^+ [2]. However, the average case complexity will be considerably lower: For many real ontologies, the number of all minimal

conflict sets for an unsatisfiable concept is much less than the size of the ontology. Our algorithm usually does not compute all the minimal conflict sets for an unsatisfiable concept. Another complexity of our algorithm comes from the computation of a minimal conflict set, which is as hard as satisfiability checking of the underlying DL. Despite the high complexity of our algorithm, fortunately, there is an optimization technique to improve its efficiency. That is, for each unsatisfiable concept to be handled, we extract a so-called syntactic locality-based module [6] from $O \cup O'$ which contains all the *minimal conflict sets* of O for C w.r.t. O'. The module extraction step can be added between line 6 and line 7 in REL_REVISION. The correctness of our modified algorithm is ensured by the fact that the locality-based module contains all the minimal sub-ontologies of an ontology that are responsible for unsatisfiability of a concept shown in in [22].

Example 2. To illustrate our iterative algorithm (i.e. Algorithm 1) based on REL_REVISION, we follow Example 1. First of all, we need to reorder all distinct confidence values in a descending order $\beta_1 = 0.93 > \beta_2 = 0.8 > \beta_3 = 0.65$ and the corresponding layers of correspondence axioms are $S_1 = \{t(m_5)\}$, $S_2 = \{t(m_3), t(m_4)\}$ and $S_3 = \{t(m_1), t(m_2)\}$ respectively. Then, we go into line 6 since $\mathcal{M}$ is inconsistent. We obtain the inconsistency degree of $\mathcal{D}$ as 0.8. So $k = 2$. As we know that $\beta_2 = 0.8$, we use $Union(\mathcal{D}_{>0.8})$ to revise S_2 and the revision result is $(S_2 \setminus \{t(m_3)\}) \cup Union(\mathcal{D}_{>0.8})$ according to REL_REVISION (see "Illustration of REL_REVISION" below). Therefore, we remove m_3 from $\mathcal{M}$ (see line 9). Then we go to another iteration of the while loop. Since the modified $\mathcal{M}$ becomes consistent when m_3 is removed from it, the whole process of Algorithm 1 can be terminated and the result is $\mathcal{D}_\star = \langle O_1, O_2, \mathcal{M} \setminus \{m_3\}\rangle$.

Illustration of REL_REVISION: The input is $O = S_2$ and $O' = Union(\mathcal{D}_{>0.8})$. Suppose the first found unsatisfiable concept is *article*. We keep on selecting the k-relevant axioms in $O \cup O'$ w.r.t. the concept *article* until $O_t = O \cup O'$ (i.e. *article* becomes unsatisfiable in O_t). Then we go to line 14 and get the minimal conflict set $\{t(m_3)\}$ of O w.r.t. O' and a hitting set $hs = \{t(m_3)\}$ (see "Illustration of CONF" below). So $HS = \{t(m_3)\}$. After this, we go to another iteration of the while loop. Since all the axioms in O have been selected, we can terminate the process and return $(S_2 \setminus \{t(m_3)\}) \cup Union(\mathcal{D}_{>0.8})$.

Illustration of CONF: The input is $C = article$, $O = S_2$ and $O' = Union(\mathcal{D}_{>0.8})$ for CONF. First of all, we compute a MCS $J = \{t(m_3)\}$ in line 3. Since only one axiom in J, we get $hs = \{t(m_3)\}$ in line 5. We return $\{t(m_3)\}$ and update $\mathcal{J} = \{t(m_3)\}$.

Neither of the above proposed revision operators removes minimal number of correspondences to resolve inconsistencies. To better fulfil minimal change, we consider the revision operator given in Algorithm 1 in [19] which utilizes a heuristics based on a *scoring function* which computes the number of minimal incoherence-preserving sub-ontologies (MIPS) that an axiom belongs to. It is not difficult to check that this revision operator satisfies conditions Inclusion, Success and Core-retainment. A MIPS of ontology O *w.r.t.* another ontology O' is a minimal sub-ontology of O that is incoherent with O'. Instantiating our

iterative algorithm with this operator results in a new conflict-based mapping revision operator. The disadvantage of this revision operator is that it needs to all the MIPSs obtained from all the minimal conflict sets of O for any concept w.r.t. O' by using a modified hitting set tree algorithm, thus its computational complexity is at least harder than those of previous revision operators.

5 Experimental Evaluation

In this section, we present the evaluation results of our algorithms by comparing them with existing algorithms. Our algorithms were implemented with the KAON2 API[1], using KAON2 as a black box reasoner. To fairly compare with the mapping repair algorithms in [13] and [15], we re-implemented them using the KAON2 API. More precisely, the following algorithms have been evaluated:

- **Weight-based-One**: Compute one minimal conflict subset each time and remove an element in it with lowest weights (see [13]).
- **Linear**: Our iterative algorithm based on the random linear base revision operator (it is equivalent to the algorithm given in [15]).
- **Weight-based-All**: Compute all minimal conflict subsets for an unsatisfiable concept and then resolve the unsatisfiability based on weights (see Algorithm 3 in [19]).
- **Relevance-based**: Our iterative algorithm based on the revision operator REL_REVISION defined in Table 1.
- **Score-based**: Our iterative algorithm based on the revision operator defined by Algorithm 1 in [19].

All of the experiments were performed on a Linux server with an Intel(R) CPU Xeon(TM) 3.2GHz running Sun's Java 1.5.0 with allotted 2GB heap space. Our system[2] including the implementation of the five algorithms can be downloaded, together with all the data sets and results.

5.1 Data sets

We use the ontology matching data sets available in OAEI'08[3] (Ontology Alignment Evaluation Initiative), which provides a platform for evaluating ontology matching systems. For our experiments, the following individual ontologies in the domain of scientific conferences are used: confOf with 197 axioms, ekaw with 248 axioms and cmt with 246 axioms. The pairwise mappings have been generated by the matching systems participating in OAEI.[4] For simplicity, we use, for example, confOf-ekaw-DSSim to indicate a distributed system consisting of individual ontologies confOf and ekaw and a mapping between them which is generated by system DSSim.

[1] http://kaon2.semanticweb.org/

[2] http://radon.ontoware.org/downloads/mapping-revision-09.zip

[3] http://oaei.ontologymatching.org/2008/

[4] In addition, we use FOAM (http://ontoware.org/projects/map/) to generate mappings by applying only some simple string-based algorithms.

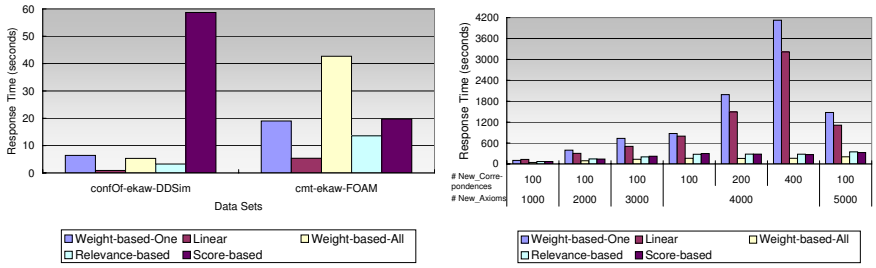

Fig. 1. The revision time of the algorithms.

5.2 Evaluation Results

We evaluated our algorithms with respect to the following measures: efficiency, scalability and correctness.

Efficiency and Scalability To measure efficiency and scalability of the algorithms, we considered revision time, which includes the time used to check whether a mapping is inconsistent and the time used to resolve inconsistencies. If a module extraction algorithm was applied, the time used to extract modules was also included. To test the efficiency of an algorithm, we continuously ran it for 30 times and took the average revision time. We have done the experiment based on the distributed systems $\mathcal{D}_1$:confOf-ekaw-DSSim and $\mathcal{D}_2$:cmt-ekaw-FOAM.

The left part of Figure 1 shows the average revision time over all runs for each algorithm. From this part we can see that for $\mathcal{D}_1$ and $\mathcal{D}_2$ which contain relatively small size of ontologies (i.e. no more than 250 axioms for each individual ontology) with mappings consisting of few correspondences (e.g. 19 and 55 correspondences for the mappings in $\mathcal{D}_1$ and $\mathcal{D}_2$ separately), Linear outperforms all the others. The second observation is that our Relevance-based outperforms all the others except Linear. It is because we expand only one branch of the hitting set tree in a depth-first manner and we apply the module extraction to optimize expanding the hitting set tree. Score-based has the worst performance for $\mathcal{D}_1$ but performs better for $\mathcal{D}_2$. This is because all correspondences in the mapping of $\mathcal{D}_1$ have the same weights so Score-based needs to compute all the minimal conflict sets for all unsatisfiable concepts in $\mathcal{D}_1$, whilst the correspondences in the mapping of $\mathcal{D}_2$ can be stratified by their weights so we only need to compute some minimal conflict sets. Another observation is that Weight-based-All does not perform well for $\mathcal{D}_2$ since it needs to compute all the minimal conflict sets for an unsatisfiable concept in each iteration.

In the right part, we show the scalability of the algorithms using the extended data sets based on $\mathcal{D}_3$:cmt-ekaw-Lily. This experiment is used to show that Linear may perform worse than some other algorithms if there are many axioms and correspondences that are not involved in the conflict sets. Here #New_Axioms means the number of axioms which are newly added to each individual ontology. #New_Correspondences indicates the number of newly added correspondences. Take the first column in the right part of Figure 1 as an example, we added 1000 dummy axioms to cmt and ekaw respectively and 100 dummy correspondences

Distributed Systems	Algorithm	Repair precision			Repair recall		
		Max(Pr_i)	Avg(Pr_i)	Min(Pr_i)	Max(Rr_i)	Avg(Rr_i)	Min(Rr_i)
confOf− ekaw− DSSim	Weight-based-One	1.00	0.73	0.55	0.89	0.71	0.56
	Linear	1.00	0.75	0.50	0.78	0.67	0.56
	Weight-based-All	1.00	0.81	0.56	0.78	0.71	0.56
	Relevance-based	0.89	0.72	0.50	0.89	0.72	0.56
	Score-based	0.86	0.86	0.86	0.67	0.67	0.67
cmt− ekaw− FOAM	Weight-based-One	1.00	0.96	0.94	0.66	0.62	0.60
	Linear	1.00	0.97	0.93	0.56	0.56	0.56
	Weight-based-All	1.00	1.00	1.00	0.70	0.66	0.64
	Relevance-based	1.00	0.98	0.93	0.58	0.56	0.56
	Score-based	1.00	1.00	1.00	0.56	0.56	0.56

Table 2. Correctness of the algorithms.

between the newly introduced concepts. Similarly we constructed other data sets by adding more axioms and correspondences. According to the right part of Figure 1, when adding more axioms and correspondences to $\mathcal{D}_3$, Weight-based-one and Linear perform worse and worse. In contrast, the other three algorithms are optimized by applying the module extraction technique and thus gain the advantage over Weight-based-one and Linear in the scalability test.

Correctness In order to measure the correctness of the algorithms, we adopted the definitions of repair precision Pr and repair recall Rr in [13]. Assume $\mathcal{M}$ is a mapping between two ontologies and $\mathcal{G}$ is the reference mapping which is created manually by domain experts. Then $\mathcal{M}^- = \mathcal{M} - \mathcal{G}$ indicates those correspondences in $\mathcal{M}$ which are not correct. The repair precision and repair recall are defined as follows:

$$\text{Repair precision} : \mathsf{Pr} = \frac{\text{removed correspondences in } \mathcal{M}^-}{\text{all removed correspondences}}$$
$$\text{Repair recall} : \quad \mathsf{Rr} = \frac{\text{removed correspondences in } \mathcal{M}^-}{|\mathcal{M}^-|}$$

This experiment is again based on $\mathcal{D}_1$ and $\mathcal{D}_2$, and we continuously ran each algorithm 30 times. For each run i (i=1,...,30), we compute the repair precision Pr_i and recall Rr_i. Table 2 shows the maximal, minimal and average repair precision and recall from all runs for each algorithm.

According to Table 2, we can see that Score-based has the highest repair precision and lowest repair recall in most cases. This shows that this algorithm best fulfils the principle of minimal change. On the other hand, since Score-based removes less correspondences, it may fail to remove some erroneous correspondences. We also noticed that Weight-based-All performs slightly better than all the others except Score-based w.r.t. both average repair precision and the average repair recall. For example, Weight-based-All reaches higher average precision (i.e. 0.81) and recall (i.e. 0.71) for $\mathcal{D}_1$ and the highest average repair precision (i.e. 1) and recall (i.e. 0.66) for $\mathcal{D}_2$. This shows that this algorithm removes more correspondences which are incorrect comparing with the results from other algorithms in most cases.

6 Conclusion and Discussion

In this paper, we discussed the problem of repairing inconsistent mappings in the distributed systems. We first defined a conflict-based mapping revision operator

and provided a representation theorem for it. We then presented an iterative algorithm for mapping revision in a distributed system based on a revision operator in DLs and showed that this algorithm results in a conflict-based mapping revision operator. We showed that the algorithm given in [15], which we call Linear, can be encoded as a special iterative algorithm. We also provided an algorithm to implement an alternative revision operator based on the relevance-based selection function given in [11] which can be optimized by a module extraction technique and considered a revision operator based on a *scoring function* in [19]. All three specific iterative algorithms have been implemented. We compared these algorithms with two other existing algorithms for mapping revision in [13] and [19]. Although our experimental results are preliminary and do not tend to be conclusive, we can still make some interesting observations:

- For most of the tests, our iterative algorithms (where Linear is equivalent to the algorithm given in [15]) performed well compared with two existing mapping revision algorithms. It is interesting to see that Linear performed quite well for all the real life data, although it performed much worse than other iterative algorithms for the scalability test.
- The iterative algorithm Score-based showed its advantage over other algorithms w.r.t. minimal change and it produced the most stable results. However, it did not perform so well for the efficiency test.
- Our iterative algorithm Relevance-based was in a good position for the correctness test. It outperformed other algorithms except Linear w.r.t efficiency and had good performance on scalability test. Thus it is a good choice to revise inconsistent mappings in those distributed systems with large scale mappings and ontologies.
- Weight-based-One performed worst for the scalability test and it does not perform well for other tests, so we suggest that it can be replaced by Linear.
- We also noticed Weight-based-All had good performance for the correctness test, although it did not perform so well for the efficiency test. So it is a good alternative to the iterative algorithms.

In the future work, we could further optimize our iterative algorithms Relevance-based and Score-based. For example, since the module extraction algorithm used to optimize our algorithms is independent on the super-concept of a subsumption entailment and thus may result in a large module, we will consider a goal-directed module extraction method that can produce a smaller module than the syntactic locality-based module. For Relevance-based, we used the relevance-based selection function in this paper because it can be applied to any DL and is the basis of some other selection functions. This selection function may select too many axioms in each iteration. Other selection functions will be considered.

Acknowledgments

Research reported in this paper was partially supported by the EU in the IST project NeOn (IST-2006-027595, `http://www.neon-project.org/`). We would

like to thank the anonymous reviewers as well as Richard Booth and Renata Wassermann for their helpful comments.

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.
2. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic EL^+. In *Proc. of KI*, pages 52–67, 2007.
3. A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003.
4. J. Euzenat and P. Shvaiko. *Ontology Matching*. Berlin; Heidelberg, Springer, 2007.
5. Peter Gardenfors. *Knowledge in Flux-Modeling the Dynamic of Epistemic States*. The MIT Press, Cambridge, Mass, 1988.
6. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: extracting modules from ontologies. In *Proc. of WWW*, pages 717–726, 2007.
7. P. Haase and G. Qi. An analysis of approaches to resolving inconsistencies in DL-based ontologies. In *Proc. of IWOD*, pages 97–109, 2007.
8. S. Ove Hansson. Reversing the levi identity. *Journal of Philosophical Logic*, 22(6):637–669, 1993.
9. S. Ove Hansson. Kernel contraction. *Journal Symbolic Logic*, 59(3):845–859, 1994.
10. Sven Ove Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Kluwer Academic Publishers, 1999.
11. Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proc. of IJCAI*, pages 454–459, 2005.
12. C. Meilicke and H. Stuckenschmidt. Applying logical constraints to ontology matching. In *Proc. of KI*, pages 99–113, 2007.
13. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Repairing ontology mappings. In *Proc. of AAAI*, pages 1408–1413, 2007.
14. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Reasoning support for mapping revision. *Journal of Logic and Computation*, 2008.
15. C. Meilicke, J. Völker, and H. Stuckenschmidt. Learning disjointness for debugging mappings between lightweight ontologies. In *Proc. of EKAW*, pages 93–108, 2008.
16. B. Nebel. Base revision operations and schemes: Semantics, representation and complexity. In *Proc. of ECAI*, pages 341–345, 1994.
17. R. Peñaloza and Baris Sertkaya. Axiom pinpointing is hard. In *Proc. of DL*, 2009.
18. G. Qi. A semantic approach for iterated revision in possibilistic logic. In *Proc. of AAAI*, pages 523–528, 2008.
19. G. Qi, P. Haase, Z. Huang, Q. Ji, J. Z. Pan, and J. Völker. A kernel revision operator for terminologies - algorithms and evaluation. In *Proc. of ISWC*, pages 419–434, 2008.
20. G. Qi, J. Z. Pan, and Qiu Ji. Extending description logics with uncertainty reasoning in possibilistic logic. In *Proc. of ECSQARU*, pages 828–839, 2007.
21. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
22. B. Suntisrivaraporn, G. Qi, Q. Ji, and P. Haase. A modularization-based approach to finding all justifications for OWL DL entailments. In *Proc. of ASWC*, pages 1–15, 2008.
23. A. Zimmermann and J. Euzenat. Three semantics for distributed systems and their relations with alignment composition. In *Proc. of ISWC*, pages 16–29, 2006.