A Conflict-based Operator for Mapping Revision*

Guilin Qi, Qiu Ji, Peter Haase

Institute AIFB, University of Karlsruhe, Germany
{gqi,qiji,pha}@aifb.uni-karlsruhe.de

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 because we give priority to ontologies when resolving inconsistencies. In this paper, we first propose a conflict-based mapping revision operator and show that it can be characterized by a logical postulate. We then provide an iterative algorithm for 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 in the ontologies. Given a distributed system which is a triple consisting of two ontologies and a mapping between them, correspondences in the mapping can

* This work is partially supported by the EU in the IST project NeOn (http://www.neon-project.org/).

have different interpretations. For example, in Distributed Description Logics (DDL) [3], a correspondence in a mapping is translated into two *bridge rules* that describes 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. propose 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 of 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 a postulate from belief revision theory [8] and show that our mapping revision operator can be characterized by it (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 if the mapping contains a large number of correspondences and the sizes of the ontologies are big because it will need a large number of satisfiability checks over a large ontology. 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 [21]. 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 [18] 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], especially to the theory of belief base revision [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 of 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 from the iterative revision algorithm given in [17] 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). 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. Suppose $\mathcal{M}$ is a set of correspondences, then $\mathcal{M}$ is a mapping between O_1 and O_2 iff for all correspondences $\langle e, e', r, \alpha \rangle \in \mathcal{M}$ we have $e \in Q(O_1)$ and $e' \in Q(O_2)$.*

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 $D = [0.0, 1.0]$. A mapping is a set of correspondences whose elements are mappable.

Definition 2. *[22] 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 are listed as follows:

$$\begin{aligned}
&\mathsf{m_1} : \langle \mathsf{crs:article}, \mathsf{ekaw:Conference_Paper}, \sqsubseteq, 0.65\rangle\\
&\mathsf{m_2} : \langle \mathsf{ekaw:Workshop_Paper}, \mathsf{crs:article}, \sqsubseteq, 0.65\rangle\\
&\mathsf{m_3} : \langle \mathsf{ekaw:Document}, \mathsf{crs:program}, \sqsubseteq, 0.80\rangle\\
&\mathsf{m_4} : \langle \mathsf{crs:program}, \mathsf{ekaw:Document}, \sqsubseteq, 0.80\rangle\\
&\mathsf{m_5} : \langle \mathsf{crs:document}, \mathsf{ekaw:Document}, \sqsubseteq, 0.93\rangle
\end{aligned}$$

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 remove some of correspondences in the mapping to restore consistency. This makes the problem of mapping repair like the problem of belief revision. Thus we call the problem of repairing mappings as mapping revision. 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 a rational 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 of 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.

Definition 7. *An* incision function *σ is a function*[1] *that for each distributed system $\langle O_1, O_2, \mathcal{M}\rangle$, we have*

(i) $\sigma(MCS_{O_1,O_2}(\mathcal{M})) \subseteq \bigcup(MCS_{O_1,O_2}(\mathcal{M}))$;
(ii) *if* $\emptyset \neq \mathcal{C} \in MCS_{O_1,O_2}(\mathcal{M})$, *then* $\mathcal{C} \cap \sigma(MCS_{O_1,O_2}(\mathcal{M})) \neq \emptyset$;
(iii) *if* $m = \langle C, C', r, \alpha\rangle \in \sigma(MCS_{O_1,O_2}(\mathcal{M}))$, *then there exists* $\mathcal{C} \in MCS_{O_1,O_2}(\mathcal{M})$ *such that* $\alpha = min\{\alpha_i : \langle C_i, C'_i, r_i, \alpha_i\rangle \in \mathcal{C}\}$.

[1] A function is often assumed to be single-valued. However, in our definition, we do not follow this assumption.

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 $\mathcal{C}$. Note that we do not assume that a function is single-valued because it is a too strong requirement that an incision function always selects the same set of correspondences to remove when it is applied to the same distributed system twice.

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:*

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

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 [19]. 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 the inconsistency degree of a distributed system is the maximum confidence value α such that the α-cut set of $\mathcal{D}$ is inconsistent under some conditions. 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, $Inc(\mathcal{D}) = 0.8$.

Proposition 1. *Given* $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$ *where at least one of* O_i $(i = 1, 2)$ *is coherent, suppose* $Inc(\mathcal{D})$ *is its inconsistency degree, then we have* $Inc(\mathcal{D}) = max\{\alpha : \mathcal{D}_{\geq\alpha} \text{ is inconsistent}\}$*.*

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.

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 O_1 or O_2 is incoherent then
3          return D
4      Rearrange the weights in M such that β_1>β_2>...>β_l > 0;
5      S_i := {t(⟨C, C', r, α⟩) : ⟨C, C', r, α⟩∈M, α = β_i}, i = 1, ..., l;
6      while M in D is inconsistent do
7          if β_k = Inc(D) then
8              S_t := S_k \ (S_k ⋆ (Union(D)_{>β_k}));
9              M := M \ {⟨C, C', r, α⟩ : t(⟨C, C', r, α⟩) ∈ S_t, α = β_k};
10     return D
11 end
```

(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 exist a concept A in O_i $(i = 1, 2)$ and a 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 $Inc(\langle O_1, O_2, \mathcal{S} \cup \{m\}\rangle)_A = \alpha$.

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

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

Unlike revision operators given in [8], our conflict-based mapping revision operator is characterized by only one postulate. This is because the definition of a conflict already gives some constrains 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]. Note that the uniformity postulates in [8] which is used to ensure that a revision operator is a single-valued function is not required for our mapping revision operator.

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 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 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 [19] 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 $O_s \cup O' \cup \{\phi\}$.

We show that the revision operator obtained by Algorithm 1 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, arbitrarily 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 $S_0 = O'$, $S_i = \{\phi_i\}$ 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 2. *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, and vice versa.*

As shown in [15], their algorithm only needs at most n satisfiability check, where n is the number of correspondences. Therefore, our iterative algorithm

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

Table 1. Relevance-based mapping revision 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 randomly ranks correspondences with the same confidence value, so its results may vary a lot if we run it for many times. Furthermore, if the size of the union of ontoloiges 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.

Definition 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 [20]. 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 [18]. 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. 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, this worst case complexity can hardly happen. 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 [21].

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 $\mathcal{J} = \{t(m_3)\}$.

5 Conclusions

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 result in a conflict-based mapping revision operator. We showed that the algorithm given in [15] 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.

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. G. Qi. A semantic approach for iterated revision in possibilistic logic. In *Proc. of AAAI*, pages 523–528, 2008.
18. 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.
19. 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.
20. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
21. 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.
22. A. Zimmermann and J. Euzenat. Three semantics for distributed systems and their relations with alignment composition. In *Proc. of ISWC*, pages 16–29, 2006.