A Kernel Revision Operator for Terminologies — Algorithms and Evaluation*

Guilin Qi[1], Peter Haase[1], Zhisheng Huang[2], Qiu Ji[1], Jeff Z. Pan[3], and Johanna Völker[1]

[1] Institute AIFB, University of Karlsruhe, Germany
{gqi,pha,qiji,jvo}@aifb.uni-karlsruhe.de
[2] Department of Mathematics and Computer Science, Vrije University Amsterdam
huang@cs.vu.nl
[3] Department of Computing Science, The University of Aberdeen
jpan@csd.abdn.ac.uk

Abstract. Revision of a description logic-based ontology deals with the problem of incorporating newly received information consistently. In this paper, we propose a general operator for revising terminologies in description logic-based ontologies. Our revision operator relies on a reformulation of the kernel contraction operator in belief revision. We first define our revision operator for terminologies and show that it satisfies some desirable logical properties. Second, two algorithms are developed to instantiate the revision operator. Since in general, these two algorithms are computationally too hard, we propose a third algorithm as a more efficient alternative. We implemented the algorithms and provide evaluation results on their efficiency, effectiveness and meaningfulness in the context of two application scenarios: Incremental ontology learning and mapping revision.

1 Introduction

Ontologies are typically not static entities, but they evolve over time and need to revised. Changes to an ontology may be caused, e.g., by modifications in the application domain, the reorganization of existing information, or the incorporation of additional knowledge according to changes in the users' needs.

An important problem in revising ontologies is maintaining the consistency of the ontology, i.e. the accommodation of new knowledge in an ontology without introducing logical contradictions. Due to the variety of sources and consequences of changes, such a revision is not a trivial process and thus cannot be left as manual work to the ontology engineer. Especially in the context of semi-automated ontology engineering, in which the ontology engineer is supported by agents (e.g. in the form of ontology learning tools) that suggest ontology changes, an automated revision is desired.

* Guilin Qi, Peter Haase, Qiu Ji and Johanna Völker are partially supported by the EU in the IST project NeOn (http://www.neon-project.org/). Jeff Z. Pan is partially supported by the EU MOST project (http://www.most-project.eu/). Zhisheng Huang is partially supported by EU-funded Projects OpenKnowledge and LarKC. We thank the reviewers for very helpful comments to improve the quality of our work.

A. Sheth et al. (Eds.): ISWC 2008, LNCS 5318, pp. 419–434, 2008.

Generally, we can distinguish two kinds of logical contradictions: inconsistency and incoherence. An ontology is inconsistent iff it has no model, i.e., it is inconsistent in the first-order sense. An ontology is incoherent iff there exists some unsatisfiable concept (i.e, an unsatisfiable concept stands for the empty set). There is a close relationship between inconsistency and incoherence [5], i.e., inconsistency is often caused by adding instances of concepts and relations to an incoherent ontology. However, an ontology can be incoherent but consistent. Incoherence is a problem that occurs in terminologies of ontologies. Resolving incoherence in a single terminology has been widely discussed (for example, see [19, 20]). However, there is very little work on resolving incoherence between terminologies of different ontologies.

There exists a number of prior work on revision in DLs, such as those reported in [5, 6, 7, 16]. Most of them focus on postulates for revision operators. For example, an important principle is that one should delete information in the original ontology as little as possible to accommodate the new knowledge consistently. Theoretically, it is important to know how to characterize a revision operator in terms of postulates. However, for practical applications, we require concrete revision operators that can be used. There are concrete revision operators defined to deal with inconsistency [7, 16]. But to the best of our knowledge, there is no revision operator dealing specifically with incoherence (as opposed to inconsistency) in the context of revision.

In this paper, we propose a kernel revision operator in Description Logic-based ontologies based on MIPS (minimal incoherence-preserving sub-terminologies) and an incision function. The notion of MIPS is originally developed for non-standard reasoning service in debugging incoherent terminologies [19, 20]. It is similar to the notion of a *kernel set* in belief base change defined in [10]. In order to resolve the logical contradiction, the incision function is used to select from each MIPS the axioms to be removed from the original ontology. Our revision operator focuses on revising terminologies, i.e. the TBox-part of ontologies. Two algorithms are developed to define specific kernel revision operators. The first algorithm is based on the reformulation of Reiter's Hitting Set Tree (HST) algorithm given in [20] and a scoring function. In this algorithm, we first compute all the MIPSs of the original ontology w.r.t. the new ontology. Then we calculate for each axiom in the MIPS a score corresponding to the number of MIPS which contain this axiom. Finally, we take subsets of those MIPSs that contain axioms with maximal scores and apply the reformulated HST algorithm to get a set of axioms to be deleted. The second algorithm is applied to ontologies where each axiom is attached a confidence value which indicates the reliability of the axiom. Such confidence values, as well as other kinds of provenance information, are typically generated by automated agents such as ontology learning or matching tools. The motivation for exploiting confidence information in this algorithm is to delete only axioms that are least reliable from each MIPS of the original ontology w.r.t. the new ontology. In this algorithm, we need to compute all the MIPSs, which is computationally hard in general. Therefore, we propose a third, alternative algorithm, which utilizes confidence values attached to axioms in the ontology to resolve unsatisfiable concepts without computing all the MIPSs. Compared to the second algorithm, this algorithm is computationally easier, but it does not necessarily remove more axioms from the original ontology after

revision. Although it does not produce a kernel revision operator, it can be viewed as a good variant of the second algorithm.

We implemented the three algorithms and provide evaluation results on their efficiency and effectiveness in the context of two application scenarios: Incremental ontology learning and mapping revision. To evaluate the scalability of our algorithms, we iteratively add a set of new terminology axioms to an ontology. We also evaluate the effectiveness of the algorithms by counting the number of axioms deleted from the old ontology by our algorithms in each iteration. Finally, we evaluate the meaningfulness of the revision results by means of a user study, where the meaningfulness is measured by the ratio of correct removals.

The rest of this paper is organized as follows: Section 2 provides a preliminary introduction to Description Logics and various notions in ontology debugging. Section 3 overviews related work of revision in DLs. Section 4 presents our revision operator for terminologies. Section 5 proposes some algorithms to instantiate the revision operator. In Section 6, we report on evaluation results with real life data. We conclude in Section 7.

2 Preliminaries

This section introduces some basic notions of Description Logics (DLs) as well as the essential notions of debugging terminologies. Since our revision operator is independent of a specific DL language, and thus can be applied to any DL, we only give a general overview of description logics.

In our work, we focus on DL-based terminological ontologies: A terminology (TBox) $\mathcal{T}$ consists of concept axioms and role axioms. A subset of a TBox is called a sub-TBox. Concept axioms have the form $C \sqsubseteq D$ where C and D are (possibly complex) concept descriptions[1], and role axioms are expressions of the form $R \sqsubseteq S$, where R and S are role descriptions. We will refer to both concept axioms and role axioms as terminology axioms.

The semantics of DLs is defined via a model-theoretic semantics, which explicates the relationship between the language syntax and the model of a domain: 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 elements of the domain, 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 $C \sqsubseteq D$ (respectively, a role inclusion axiom $R \sqsubseteq S$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, respectively). An interpretation $\mathcal{I}$ is called a *model* of a TBox $\mathcal{T}$, iff it satisfies each axiom in $\mathcal{T}$. We use $Mod(\mathcal{T})$ to denote all the models of a TBox $\mathcal{T}$. A named concept C in a terminology $\mathcal{T}$ is unsatisfiable iff, for each model $\mathcal{I}$ of $\mathcal{T}$, $C^{\mathcal{I}} = \emptyset$. A terminology $\mathcal{T}$ is incoherent iff there exists an unsatisfiable named concept in $\mathcal{T}$. Two TBoxes $\mathcal{T}$ and $\mathcal{T}'$ are equivalent, denoted by $\mathcal{T} \equiv \mathcal{T}'$, iff $Mod(\mathcal{T}) = Mod(\mathcal{T}')$.

We now introduce the notions of MIPS and MUPS which will be used to define our revision operator. Both of these terms have originally been defined in [19] and are used to pinpoint errors in an ontology.

[1] A complex concept is a concept that is formed by some atomic concepts and constructors such as conjunction $\sqcap$ and disjunction $\sqcup$.

Definition 1. *Let A be a named concept which is unsatisfiable in a TBox $\mathcal{T}$. A set $\mathcal{T}' \subseteq \mathcal{T}$ is a* minimal unsatisfiability-preserving sub-TBox (MUPS) *of $\mathcal{T}$ w.r.t. A if A is unsatisfiable in $\mathcal{T}'$, and A is satisfiable in every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$.*

A MUPS of $\mathcal{T}$ *w.r.t.* A is a minimal sub-TBox of $\mathcal{T}$ in which A is unsatisfiable.

Example 1. Let $\mathcal{T} = \{A \sqsubseteq B, A \sqsubseteq \neg B, C \sqsubseteq A, C \sqsubseteq D, C \sqsubseteq \neg D\}$. There are two unsatisfiable concepts in $\mathcal{T}$: A and C. It is easy to check that there are two MUPSs of $\mathcal{T}$ w.r.t. C: $\{A \sqsubseteq B, A \sqsubseteq \neg B, C \sqsubseteq A\}$ and $\{C \sqsubseteq D, C \sqsubseteq \neg D\}$, and there is one MUPS of $\mathcal{T}$ w.r.t. A: $\{A \sqsubseteq B, A \sqsubseteq \neg B\}$.

MUPSs are useful for relating sets of axioms to the unsatisfiability of specific concepts, but they can also be used to calculate minimal incoherence preserving sub-TBoxes, which relate sets of axioms to the incoherence of a TBox in general and are defined as follows.

Definition 2. *Let $\mathcal{T}$ be an incoherent TBox. A TBox $\mathcal{T}' \subseteq \mathcal{T}$ is a* minimal incoherence-preserving sub-TBox (MIPS) *of $\mathcal{T}$ if $\mathcal{T}'$ is incoherent, and every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$ is coherent.*

A MIPS of $\mathcal{T}$ is the minimal sub-TBox of $\mathcal{T}$ which is incoherent. For $\mathcal{T}$ in Example 1, we get the following MIPSs: $\{A \sqsubseteq B, A \sqsubseteq \neg B\}$ and $\{C \sqsubseteq D, C \sqsubseteq \neg D\}$.

3 Related Work and Motivation

This work is related to belief revision which has been widely discussed in the literature. The theory of belief revision in propositional and first-order logic deals with logical inconsistency resulting from revising a knowledge base by newly received information. Alchourrón, Gardenfors and Markinson (AGM for short) [1] propose a set of postulates to characterize a revision operator. In AGM's work, beliefs of an agent are represented by a set of formulas closed under logical consequence, called a belief set. A revision operator is an operation that maps a belief set and a formula to a belief set. This representation is afflicted by a number problems. For example, there is potentially an infinite number of formulas in a belief set. Therefore, several researchers have proposed to use a belief base which is a set of formulas that is not closed under logical consequence to represent the beliefs of an agent [11, 13]. In the scenario of ontology change, this later representation seems to be more natural because we do not require that an ontology should be closed under logical consequence.

The problem of revision in DLs has been extensively studied in the literature. In [6], Flouris, Plexousakis and Antoniou generalize the AGM framework in order to apply the rationales behind the AGM framework to a wider class of logics, i.e. a larger class of logics which are AGM-compliant. In [5], a framework for the distinction between incoherence and inconsistency of an ontology is proposed. A set of rational postulates for a revision operator in DLs is proposed based on the distinction between coherent negation and consistent negation. However, in [5] no concrete revision operator is proposed. In [16], reformulated AGM postulates for revision are adapted to DLs. Two revision operators that satisfy the adapted postulates are given, but no algorithm to implement any of the operators is introduced.

Similar to our revision operator, the revision operator defined in [18] also utilizes an *incision function* to select axioms to be removed from the original ontology. Our work differs from theirs in several aspects. First, our revision operator deals with incoherence instead of inconsistency. Second, we provide algorithms for computation of specific revision operators and discuss evaluation results on their implementation. This work is also related to the work presented in [8], in which an algorithm is given to determine consistent sub-ontologies by adding an axiom to an ontology. The algorithm is based on a selection function by assuming that all axioms in the ontology are connected. Recently, a revision operator has been defined to repair erroneous mappings derived by automated ontology alignment systems [14]. Their revision operator, however, calculates neither MUPS nor MIPS and may remove too much information.

According to the discussion of related work, although there is no revision operator dealing with incoherence, it is possible to define such a revision operator based on the result of debugging and diagnosis. However, there are several problems to be solved. First, the notions of MUPS and MIPS are defined on a single ontology, whilst we need to consider two ontologies such that one of them is more important than the other. Therefore, we need to generalize the notions of MUPS and MIPS. Second, it has been shown in [20] that finding all the MUPS and MIPS in $\mathcal{ALC}$ is time-consuming and efficiency is a problem that prevents us from calculating all the MUPS and MIPS. This problem is even more serious for more expressive DLs (thus computationally harder). Third, even if we can find an efficient algorithm for calculating all the MIPS, we must find an efficient way to remove as few axioms as possible to restore coherence. We tackle these problems by first defining a generalized MIPS and a general revision operator based on it. We then give three algorithms to instantiate the general revision operator.

4 Kernel Revision Operator for Terminologies

In this section, we define our revision operator based on the notion of MIPS. Originally, the notion of a MIPS is defined on a single TBox, whereas a revision operator deals with conflicts between two TBoxes. We therefore generalize MIPS by considering two TBoxes: the TBox $\mathcal{T}$ to be revised, and the newly received TBox $\mathcal{T}_0$. In the following, we further assume that both $\mathcal{T}$ and $\mathcal{T}_0$ are coherent.

Definition 3. *Let $\mathcal{T}$ and $\mathcal{T}_0$ be two TBoxes. A minimal incoherence-preserving sub-TBox (MIPS) $\mathcal{T}'$ of $\mathcal{T}$ $w.r.t.$ $\mathcal{T}_0$ is a sub-TBox of $\mathcal{T}$ which satisfies (1) $\mathcal{T}' \cup \mathcal{T}_0$ is incoherent; (2) $\forall \mathcal{T}'' \subset \mathcal{T}'$, $\mathcal{T}'' \cup \mathcal{T}_0$ is coherent. We denote the set of all MIPSs of $\mathcal{T}$ $w.r.t$ $\mathcal{T}_0$ by $MIPS_{\mathcal{T}_0}(\mathcal{T})$.*

A MIPS of TBox $\mathcal{T}$ $w.r.t.$ TBox $\mathcal{T}_0$ is a minimal sub-TBox of $\mathcal{T}$ that is incoherent with $\mathcal{T}_0$. This definition of MIPS is similar to the notion of a minimal axiom set given in [2] where an ontology is split into a *static part* and a *rebuttal part*. It can be considered as the *kernel* defined by Hansson in [10]. Similar to Definition 3, we can define a MUPS of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ and an unsatisfiable concept of $\mathcal{T} \cup \mathcal{T}_0$. When $\mathcal{T}_0$ is an empty set, then Definition 3 is reduced to Definition 2. In classical logic, given a knowledge base A which is a set of classical formulas and a formula ϕ, a ϕ-kernel of A is the minimal subbase of A that implies ϕ. To define a contraction function for removing knowledge

from a knowledge base, called kernel contraction, Hansson defines an incision function which selects formulas to be discarded in each ϕ-kernel of A. We adapt the incision function to define our revision operator.

Definition 4. *Let $\mathcal{T}$ be a TBox. An* incision function *for $\mathcal{T}$, denoted as σ, is a function ($\sigma : 2^{2^{\mathcal{T}}} \rightarrow 2^{\mathcal{T}}$) such that for each TBox $\mathcal{T}_0$*

(i) $\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) \subseteq \bigcup_{\mathcal{T}_i \in MIPS_{\mathcal{T}_0}(\mathcal{T})} \mathcal{T}_i$;
(ii) if $\mathcal{T}' \in MIPS_{\mathcal{T}_0}(\mathcal{T})$, *then* $\mathcal{T}' \cap \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) \neq \emptyset$.

An incision function for a TBox $\mathcal{T}$ is a function such that for each TBox $\mathcal{T}_0$, it selects formulas from every MIPS of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ if this MIPS is not empty. Condition (i) says the axioms selected by an incision function must belong to some MIPSs of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$. Condition (ii) says each MIPS of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ must have at least one axiom selected. The incision function plays a similar role as concept pinpointing in [19]. However, the latter is only applied to a single ontology.

An important incision function is the one which is called *minimal incision function* [4]. The idea of this incision function is to select a minimal subset of elements from the set of kernel sets. We adapt this incision function as follows.

Definition 5. *Let $\mathcal{T}$ be a TBox. An incision function σ for $\mathcal{T}$ is* minimal *if there is no other incision function σ' for $\mathcal{T}$ such that there is a TBox $\mathcal{T}_0$, $\sigma'(MIPS_{\mathcal{T}_0}(\mathcal{T})) \subset \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T}))$.*

A minimal incision function selects a minimal subset of $\mathcal{T}$ w.r.t. the set inclusion. However, among all the minimal incision functions, some of them select more axioms than others. To make the number of selected axioms minimal, we define a cardinality-minimal incision function.

Definition 6. *Let $\mathcal{T}$ be a TBox. An incision function σ for $\mathcal{T}$ is* cardinality-minimal *if there is no other incision function σ' such that there is a TBox $\mathcal{T}_0$, $|\sigma'(MIPS_{\mathcal{T}_0}(\mathcal{T}))| < |\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T}))|$.*

It is clear that a cardinality-minimal incision function is always a minimal incision function.

Proposition 1. *Let $\mathcal{T}$ be a TBox. Suppose σ is a cardinality-minimal incision function for $\mathcal{T}$, then it is a minimal incision function.*

From each incision function, we can define an operator for revising a TBox $\mathcal{T}$ by a newly received TBox $\mathcal{T}_0$. The idea is that we first calculate the MIPS of TBox $\mathcal{T}$ $w.r.t$ TBox $\mathcal{T}_0$, then delete axioms in $\mathcal{T}$ selected by the incision function. After that, we take the union of the modified TBox and $\mathcal{T}_0$ as the result of the revision.

Definition 7. *Let $\mathcal{T}$ be a TBox, and σ be an incision function for $\mathcal{T}$. The* kernel revision *operator $\circ_\sigma$ for $\mathcal{T}$ is defined as follows: for each TBox $\mathcal{T}_0$,*

$$\mathcal{T} \circ_\sigma \mathcal{T}_0 = (\mathcal{T} \setminus \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T}))) \cup \mathcal{T}_0.$$

The result of a revision by the kernel revision operator only contains a single TBox. According to the definition of an incision function, the resulting TBox of the kernel revision operator is always a unique coherent TBox.

Proposition 2. *Let $\mathcal{T}$ be a TBox, and σ be an incision function for $\mathcal{T}$. The operator $\circ_\sigma$ satisfies the following properties: for any TBoxes $\mathcal{T}_0$, $\mathcal{T}_0'$*
(R1) $\mathcal{T}_0 \subseteq \mathcal{T} \circ_\sigma \mathcal{T}_0$.
(R2) If $\mathcal{T} \cup \mathcal{T}_0$ is coherent, then $\mathcal{T} \circ_\sigma \mathcal{T}_0 = \mathcal{T} \cup \mathcal{T}_0$.
(R3) If $\mathcal{T}_0$ is coherent, then $\mathcal{T} \circ_\sigma \mathcal{T}_0$ is coherent.
(R4) If $\mathcal{T}_0 \equiv \mathcal{T}_0'$, then $\mathcal{T} \circ_\sigma \mathcal{T}_0 \equiv \mathcal{T} \circ_\sigma \mathcal{T}_0'$.
(R5) If $\phi \in \mathcal{T}$ and $\phi \notin \mathcal{T} \circ_\sigma \mathcal{T}_0$, then there is a subset S of $\mathcal{T}$ and a subset S_0 of $\mathcal{T}_0$ such that $S \cup S_0$ is coherent, but $S \cup S_0 \cup \{\phi\}$ is not.

Proof. (sketch) It is clear that (R1)-(R3) hold. We show that (R4) holds. Suppose $\mathcal{T}_0 \equiv \mathcal{T}_0'$. According to Definition 3, we must have $MIPS_{\mathcal{T}_0}(\mathcal{T}) = MIPS_{\mathcal{T}_0'}(\mathcal{T})$. Therefore, we have $\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) = \sigma(MIPS_{\mathcal{T}_0'}(\mathcal{T}))$. It follows that $\mathcal{T} \circ_\sigma \mathcal{T}_0 \equiv \mathcal{T} \circ_\sigma \mathcal{T}_0'$. (R5) holds because the incision function that is used to define a kernel revision operator only selects axioms from MIPSs of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$. Therefore, these axioms must be in a subset of $\mathcal{T}$ that is in conflict with some axioms in $\mathcal{T}_0$.

Properties (R1)-(R4) are adapted from postulates (O+1), (O+2*), (O+3*) and (O+4) in [5]. (R1) says that every axiom in the new TBox should be accepted after revision. (R2) says, if two TBoxes have no contradiction, then we do not need to change anything. (R3) means that if the new TBox is coherent, then the result of revision should also be coherent. (R4) is a weakened form of syntax-independence. That is, the revision operator is independent of the syntactic form of axioms in the new TBox. (R5) is a new property which is adapted from the core-retainment postulate in [10]. It states that if an axiom is deleted after revision, then it must be responsible for the conflict.

5 Algorithms

The kernel revision operator is defined by an incision function. However, we have not given any incision function up to now. In the following, inspired by the work reported in [21], we propose some algorithms for computing an incision function based on Reiter's Hitting Set Tree (HST) algorithm [17] which is reformulated in [20]. We briefly introduce Reiter's theory. Given a *universal set* U, and a set $S = \{s_1, ..., s_n\}$ of subsets of U which are *conflict sets*, i.e. subsets of the system components responsible for the error. A *hitting set* T for S is a subset of U such that $s_i \cap T \neq \emptyset$ for all $1 \leq i \leq n$. A *minimal hitting set* T for S is a hitting set such that no $T' \subset T$ is a hitting set for S. Reiter's algorithm is used to calculate minimal hitting sets for a collection $S = \{s_1, ..., s_n\}$ of sets by constructing a labeled tree, called a Hitting Set Tree (HST). We select one arbitrary minimal hitting set of $MIPS_{\mathcal{T}_0}(\mathcal{T})$ given by HST algorithm in [20]. We denote the revised HST algorithm as $HSTree$. We do not apply the revised HST algorithm to $MIPS_{\mathcal{T}_0}(\mathcal{T})$ because there may have a large number of hitting sets if we use all the MIPSs and the algorithm will be very slow. Instead, we apply the revised HST algorithm to the set of subsets of the MIPSs in $MIPS_{\mathcal{T}_0}(\mathcal{T})$.

The first algorithm is based on the *scoring function on axioms*[2] which is defined as follows.

[2] A scoring function has been used in [15] to measuring inconsistency in a single ontology and is defined by MIPS, whilst ours is not defined by MIPS.

Algorithm 1. Algorithm for Repair based on scoring function

Data: Two TBoxes $\mathcal{T}$ and $\mathcal{T}_0$, where $\mathcal{T}$ is the TBox to be revised
Result: A repaired coherent TBox $\mathcal{T} \circ_\sigma \mathcal{T}_0$
begin
  $\mathcal{C} = \emptyset$
  calculate $MIPS_{\mathcal{T}_0}(\mathcal{T})$
  for $ax \in \bigcup_{\mathcal{T}_i \in MIPS_{\mathcal{T}_0}(\mathcal{T})} \mathcal{T}_i$ **do**
    $w_{ax} := S_{MIPS_{\mathcal{T}_0}(\mathcal{T})}(\{ax\})$
  for $\mathcal{T}_i \in MIPS_{\mathcal{T}_0}(\mathcal{T})$ **do**
    $A_i := \{ax \in \mathcal{T}_i : \nexists ax' \in \mathcal{T}_i, w_{ax'} > w_{ax}\}$
    $\mathcal{C} := \mathcal{C} \cup \{A_i\}$
  $\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) := HSTree(\mathcal{C})$
  $\mathcal{T} \circ_\sigma \mathcal{T}_0 := \mathcal{T} \setminus \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) \cup \mathcal{T}_0$
  return $\mathcal{T} \circ_\sigma \mathcal{T}_0$
end

Definition 8. *Let $\mathcal{T}$ be a TBox and $\mathcal{M}$ be a set of sub-TBoxes of $\mathcal{T}$. The scoring function for $\mathcal{T}$ w.r.t. $\mathcal{M}$, is a function $S_{\mathcal{M}} : \mathcal{P}(\mathcal{T}) \mapsto N$ such that for all $\mathcal{T}' \in \mathcal{P}(\mathcal{T})$*

$$S_{\mathcal{M}}(\mathcal{T}') = |\{\mathcal{T}_i \in \mathcal{M} : \mathcal{T}_i \cap \mathcal{T}' \neq \emptyset\}|.$$

The scoring function $S_{\mathcal{M}}$ for $\mathcal{T}$ returns for each subset $\mathcal{T}'$ of $\mathcal{T}$ the number of elements of $\mathcal{M}$ that have an overlap with $\mathcal{T}'$. If we apply the scoring function to each singleton $\{ax_i\}$, where ax_i is an axiom in $\mathcal{T}$, we can attach a degree to each axiom in $\mathcal{T}$.

In Algorithm 1, we first calculate all the MIPSs of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ (MIPSs for short). The approach for calculating all the MIPSs is based on a black-box algorithm for finding all justifications proposed in [12]. We then compute the score of each axiom in the union of the MIPSs (see the first "for" loop) by applying the scoring function $S_{MIPS_{\mathcal{T}_0}}$. For each MIPS, we select a subset of it containing those axioms whose scores are maximal among all the axioms in the MIPS, and we apply the modified HST algorithm to these axioms (see the second "for" loop and the line after it). The result of the modified HST algorithm is the set of axioms to be deleted, i.e., $\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T}))$. After removing them, we restore coherence of the TBox $\mathcal{T}$ w.r.t. $\mathcal{T}_0$. In our algorithm, we use subsets of MIPSs consisting of those axioms with highest scores as an input to the HST algorithm, instead of using all the MIPSs. So, the number of removed axioms may not be minimal.

Example 2. Suppose that we have two TBoxes:

$\mathcal{T} = \{E \sqsubseteq B, D \sqsubseteq \neg B, F \sqsubseteq B, F \sqsubseteq C\}$, and $\mathcal{T}_0 = \{D \sqsubseteq E, G \sqsubseteq D, F \sqsubseteq D, H \sqsubseteq A\}$. The MIPSs of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ are $\mathcal{T}' = \{E \sqsubseteq B, D \sqsubseteq \neg B\}$ and $\mathcal{T}'' = \{D \sqsubseteq \neg B, F \sqsubseteq B\}$.

The score of the disjointness axiom $D \sqsubseteq \neg B$ is 2, because it belongs to both MIPSs. The scores of the other axioms are 1. Therefore, $\mathcal{C} = \{\{D \sqsubseteq \neg B\}, \{D \sqsubseteq \neg B\}\}$ and $\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) = \{D \sqsubseteq \neg B\}$. So we delete $D \sqsubseteq \neg B$ and the result of revision is $\mathcal{T} \circ_\sigma \mathcal{T}_0 = \{E \sqsubseteq B,\ F \sqsubseteq B,\ F \sqsubseteq C, D \sqsubseteq E, G \sqsubseteq D, F \sqsubseteq D, H \sqsubseteq A\}$.

In some cases, there are confidence values attached to axioms in an ontology. These confidence values can be generated during an ontology learning process (see [9]) or

Algorithm 2. Algorithm for Repair based on confidence values

Data: Two TBoxes $\mathcal{T}$ and $\mathcal{T}_0$, where $\mathcal{T}$ is the TBox to be revised, each axiom ax in $\mathcal{T}$ is attached a confidence value w_{ax}
Result: A repaired coherent TBox $\mathcal{T} \circ_c \mathcal{T}_0$
begin
  $\mathcal{C} = \emptyset$
  calculate $MIPS_{\mathcal{T}_0}(\mathcal{T})$
  for $\mathcal{T}_i \in MIPS_{\mathcal{T}_0}(\mathcal{T})$ **do**
    $A_i := \{ax \in \mathcal{T}_i : \nexists ax' \in \mathcal{T}_i, w_{ax'} < w_{ax}\}$
    $\mathcal{C} := \mathcal{C} \cup \{A_i\}$
  $\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) := HSTree(\mathcal{C})$
  $\mathcal{T} \circ_c \mathcal{T}_0 := \mathcal{T} \setminus \sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) \cup \mathcal{T}_0$
  return $\mathcal{T} \circ_c \mathcal{T}_0$
end

given by human experts. When confidence values are attached to axioms in the TBox, we can choose an axiom with least confidence from a MIPS and delete it. Note that we do not need to know the exact values attached to the axioms. What matters is the total ordering between axioms. A natural idea is to replace the score w_{ax} of each axiom ax in Algorithm 1 by its confidence value if applicable. This leads us to Algorithm 2.

In Algorithms 1 and 2, we extract a subset of each MIPS which consists of either those axioms with the maximal score or those with least confidence values. We then apply the modified HST algorithm to these subsets to find a hitting set. Our algorithms clearly compute an incision function, which is not the minimal incision function. However, according to our experiment on real life ontologies, our algorithms delete only a small number of axioms in order to restore consistency and have acceptable evaluation of meaningfulness.

Example 3. (Example 2 Continued) Suppose axioms in the TBox $\mathcal{T}$ are attached with confidence values as follows:
$w_{E \sqsubseteq B} = 0.4$, $w_{D \sqsubseteq \neg B} = 0.5$, $w_{F \sqsubseteq B} = 0.6$, $w_{F \sqsubseteq C} = 0.9$.
The axioms in $\mathcal{T}_0$ are assigned weight 1, i.e., they are firmly believed.

It is clear that $\mathcal{A}' = \{E \sqsubseteq B\}$ and $\mathcal{A}'' = \{D \sqsubseteq \neg B\}$. Therefore, $\mathcal{C} = \{\{E \sqsubseteq B\}, \{D \sqsubseteq \neg B\}\}$ and $\sigma(MIPS_{\mathcal{T}_0}(\mathcal{T})) = \{E \sqsubseteq B, D \sqsubseteq \neg B\}$. Therefore, $\mathcal{T} \circ_c \mathcal{T}_0 = \{F \sqsubseteq B, G \sqsubseteq D, D \sqsubseteq E, H \sqsubseteq A, F \sqsubseteq C, F \sqsubseteq D\}$.

In Algorithm 3, when resolving incoherence of a TBox, we do not compute all the MIPSs. Instead, we resolve incoherence by iteratively dealing with unsatisfiable concepts. That is, we remove axioms in the MUPSs of an unsatisfiable concept and make it satisfiable before dealing with another unsatisfiable concept, and so on. The function which computes all the MUPSs of $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ and C is similar to the algorithm to compute MUPS in [12], and it is denoted by $GETMUPS_{\mathcal{T}_0}(C, \mathcal{T})$. The only difference is that after computing a single MUPS of $\mathcal{T} \cup \mathcal{T}_0$ w.r.t. C, we only take the intersection of the MUPS and $\mathcal{T}$ as the node in the Hitting Set Tree. For each unsatisfiable concept, we take the subset of every MUPS which contains axioms with minimal confidence values and then apply the HST algorithm to select the axioms to be deleted. In this sense, this algorithm still achieves some kind of minimal change when resolving unsatisfiability of

Algorithm 3. Adapted algorithm for Repair based on confidence values

Data: Two TBoxes $\mathcal{T}$ and $\mathcal{T}_0$, where $\mathcal{T}$ is the TBox to be revised, axioms in $\mathcal{T}$ are attached with confidence values
Result: A repaired coherent TBox $\mathcal{T} \circ_w \mathcal{T}_0$
begin
  $\mathcal{C} := \emptyset$
  for $C \in GETALLCONCEPTS(\mathcal{T} \cup \mathcal{T}_0)$ **do**
    while $\mathcal{T} \cup \mathcal{T}_0 \models C \sqsubseteq \bot$ **do**
      $\mathcal{M}_{C,\mathcal{T},\mathcal{T}_0} := GETMUPS_{\mathcal{T}_0}(C, \mathcal{T})$
      for $\mathcal{T}_i \in \mathcal{M}_{C,\mathcal{T},\mathcal{T}_0}$ **do**
        $\underline{\mathcal{T}_i} := \{ax \in \mathcal{T}_i : \nexists ax' \in \mathcal{T}_i, w_{ax'} < w_{ax}\}$
        $\mathcal{C} := \mathcal{C} \cup \{\underline{\mathcal{T}_i}\}$
      $\mathcal{T}_C := HSTree(\mathcal{C})$
      $\mathcal{T} := \mathcal{T} \setminus \mathcal{T}_C$
      $\mathcal{C} := \emptyset$
  return $\mathcal{T} \cup \mathcal{T}_0$
end

a concept, even if the revision operator implemented by this algorithm is not a kernel revision operator. As we we do not need to calculate all the MIPSs, the algorithm is much more efficient than Algorithm 2 as long as all the MIPS in Algorithms 1 and 2 are calculated from all the MUPS as suggested by Schlobach and Cornet in [19].

Example 4. (Example 3 Continued) There are three unsatisfiable concepts in $\mathcal{T} \cup \mathcal{T}_0$: G, D and F. Suppose our algorithm chooses F first. The MUPS of F in $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ is $\mathcal{T}' = \{D \sqsubseteq \neg B, F \sqsubseteq B\}$. So $\mathcal{M}_{F,\mathcal{T},\mathcal{T}_0} = \{\mathcal{T}'\}$. Since $w_{D \sqsubseteq \neg B} < w_{F \sqsubseteq B}$, we have $\mathcal{C} = \{\{D \sqsubseteq \neg B\}\}$. So $\mathcal{T}_\mathcal{C} = \{D \sqsubseteq \neg B\}$.

We replace $\mathcal{T}$ by $\mathcal{T} \setminus \{D \sqsubseteq \neg B\}$. It is easy to check that $\mathcal{T} \cup \mathcal{T}_0$ is coherent now. So, the algorithm terminates and the result of the revision is $\mathcal{T} \circ_w \mathcal{T}_0 = \{F \sqsubseteq B, E \sqsubseteq B, G \sqsubseteq D, D \sqsubseteq E, H \sqsubseteq A, F \sqsubseteq C, F \sqsubseteq D\}$.

6 Experimental Evaluation

Our algorithms have been implemented in Java as part of the RaDON plugin[3] for the NeOn Toolkit.[4] In this section, we provide an evaluation and comparison of the algorithms with respect to efficiency, effectiveness and meaningfulness. The experiments have been performed on a Linux server running Sun's Java 1.5.0 with a maximum heap space 2048 MB. For each revision operation, the maximal time limit is 1 hour.

6.1 Application Scenarios and Data Sets

We performed the evaluation in an ontology learning scenario and an ontology mapping scenario. All data sets can be downloaded from RaDON website[5]. In the ontology

[3] http://radon.ontoware.org/
[4] http://www.neon-toolkit.org/
[5] http://radon.ontoware.org/downloads/data-revision-iswc08.zip

learning scenario, an ontology is automatically and incrementally generated using ontology learning algorithms. Dealing with incoherence is especially important in ontology learning: Due to the nature of ontology learning algorithms, the acquired ontologies inherently represent uncertain and possibly contradicting knowledge. In the ontology mapping scenario, we start with two heterogeneous source ontologies, which are then extended and revised by adding mappings relating elements of the two ontologies. The mappings are created by an ontology matching system. As in the case of ontology learning, also the matching systems produce uncertain and potentially erroneous mappings. As a result, the integrated ontologies become incoherent in many cases. Resolving the incoherence caused by the mappings is a critical task to improve the quality of ontology mapping results.

Ontology learning scenario: We applied the ontology learning framework Text2Onto[6] on a text corpus consisting of abstracts from the "knowledge management" information space of the BT Digital Library. We extracted concepts, taxonomic and non-taxonomic relationships, as well as disjointness axioms from the documents in the information space. The generated axioms are annotated with confidence values based, e.g., on lexical context similarity or the frequency of lexico-syntactic patterns matched in the text. The generated ontology bt_km comprises $4,000$ terminological axioms in total.

Starting with an initially empty ontology, in every revision step we incrementally[7] add an ontology $\mathcal{T}_0$ of 100 randomly generated axioms to the ontology $\mathcal{T}$. For each iteration, if $\mathcal{T}$ w.r.t. $\mathcal{T}_0$ turns incoherent, we apply our algorithms to obtain a coherent revised ontology. Otherwise, we simply add $\mathcal{T}_0$ to $\mathcal{T}$. Then the revised ontology (i.e. the modified $\mathcal{T}$) serves as input for the next iteration.

Ontology mapping scenario: Here we address the scenario of integrating two heterogeneous source ontologies via mappings. While the individual source ontologies are locally coherent, relating them with mapping axioms may turn the integrated ontologies globally incoherent. In this scenario, we assume the two source ontologies to be fixed and the generated mappings to be revised in the case of logical contradictions. Therefore, we apply our revision algorithm to remove only mapping axioms and treat the source ontology axioms as stable.

For this scenario, we use the ontology mapping data sets provided by the University of Mannheim.[8] The data sets include some source ontologies and mappings used in the ontology alignment evaluation initiative[9], which provides a platform to evaluate ontology matching systems. For our test, we use as source ontologies different ontologies about the domain of scientific conferences: *CONFTOOL* (a $\mathcal{SIF}(\mathcal{D})$ ontology), *CMT* (a $\mathcal{ALCIF}(\mathcal{D})$ ontology), *EKAW* (a $\mathcal{SHIN}$ ontology), *CRS* (a DL-Lite ontology) and *SIGKDD* (a $\mathcal{ALI}(\mathcal{D})$ ontology) with 197, 246, 248, 69 and 122 axioms respectively. The pairwise mappings were generated automatically by the HMatch system [3]. They

[6] http://ontoware.org/projects/text2onto/

[7] By adding set of axioms incrementally, we are actually doing iterated revision. The purpose of doing this is to evaluate the scalability of our algorithms. Discussions on iterated revision using our revision operators are out of the scope of this paper and will be left as future work.

[8] http://webrum.uni-mannheim.de/math/lski/ontdebug/index.html

[9] http://om2006.ontologymatching.org/

include *CONFTOOL-CMT* with 14 mapping axioms, *EKAW-CMT* with 46 mapping axioms and *CRS-SIGKDD* with 22 mapping axioms. We selected these ontologies and mappings for our experiments because they exhibit inconsistencies when integrated, and are thus interesting for revision experiments.

6.2 Evaluation Measures

In the following, we evaluate our algorithms with respect to aspects of efficiency, effectiveness and meaningfulness.

For measuring its efficiency, we provide the revision time t including the time to check whether the ontology is incoherent as well as the time to debug and resolve the incoherence. We further measure the effectiveness of our algorithms in terms of the number R of axioms which need to be removed from $\mathcal{T}$ to restore the coherence. The fewer axioms are removed, the better the algorithm complies with the principle of minimal change.

In order to measure the meaningfulness of our algorithms, four users are asked to assess whether the removal of an axiom in a particular revision was correct from their point of view. Specifically, we provide several axioms which are selected for removal by our algorithms as well as the MIPSs and MUPSs containing them (and scores of the axioms or confidence degrees attached to the axioms if applicable). For each removed axiom, we ask the users to decide whether the removal: (1) was correct, (2) was incorrect, or (3) whether they are unsure. For the evaluated results returned by each user, the meaningfulness is then measured by the ratio of correct removals:

$$\text{Correctness} = \frac{\#Correct\_Removals}{\#Total\_Removals}$$

Similarly we can define an "Error_Rate" based on the incorrect removals and an "Unknown_Rate" based on the removals where the users were unsure. We combine the obtained Correctness (respectively Error_Rate and Unknown_Rate) values from different users by averaging them.

6.3 Evaluation Results

Analysis of Efficiency and Effectiveness

Results for the ontology learning scenario: The runtime performance of our algorithms over ontology bt_km is depicted by Figure 1. Additional details for the entire ontology ($4,000$ axioms) are shown in Table 1. It can be seen from the figure that the accumulative revision time does not linearly increase with the number of TBox axioms. This is because the revision time is related not only to the size of the input TBox, but also to the number of MUPSs. Take the iteration when the size of the current TBox $\mathcal{T}$ reaches about $2,900$ as an example. In this iteration, Algorithm 1 computes 124 unsatisfiable concepts and 154 MUPSs based on its previous revision results. The MUPSs found in this iteration are much more than those obtained in previous iterations, and thus the accumulative time for Algorithm 1 increases sharply in such case. This explanation can be also applied to Algorithm 2, while for Algorithm 3 in this iteration, only 3 unsatisfiable

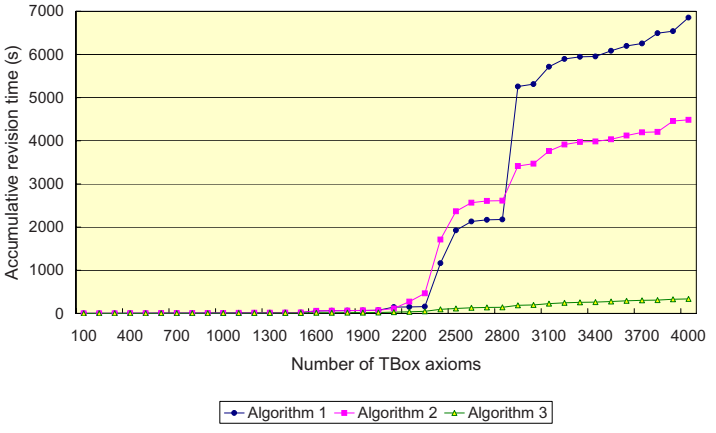

Fig. 1. The runtime performance of ontology revision for ontology bt_km

Table 1. Accumulative results for the entire ontology bt_km

Algorithm	# of Unsatisf. Concepts	# of MUPS	# of Removals	Revision time t (Seconds)
Algorithm 1	581	790	27	6,856
Algorithm 2	506	717	34	4,486
Algorithm 3	30	39	33	338

concepts and 3 MUPSs are computed, similarly to the previous iterations. Thus in this iteration, the accumulative time for this algorithm increases smoothly.

Let us now compare Algorithm 1 with Algorithm 2, since they share the same procedure to debug incoherence[10], but apply different strategies to resolve incoherence: Table 1 shows that Algorithm 1 removes fewer axioms than Algorithm 2, while taking more time to revise. On the one hand, the strategy using a scoring function better follows the principle of minimal change. On the other hand, as Algorithm 2 removes more axioms, more potential incoherence is resolved which may exist when new information is added. Thus, less MUPSs are computed in the end.

Second, we compare Algorithm 2 with Algorithm 3, because they use the same strategy to restore coherence while relying on different debug procedures. Algorithm 3 is considerably faster than Algorithm 2, since less MUPSs need to be computed. From the MUPSs obtained by the two algorithms, we observe that most of the unsatisfiable concepts can be derived from others. In such case, if we resolve the unsatisfiability of some concepts, others will be resolved automatically. Therefore, Algorithm 3 is much more efficient than Algorithm 2.

Results for the ontology mapping scenario: Table 2 presents the evaluation results of our algorithms based on the mapping data set described above. According to table 2, Algorithm 3 outperforms the other two w.r.t. efficiency. The reason is that Algorithm 3

[10] When we say debug incoherence, we mean finding all the MUPSs of an unsatisfiable concept or finding all MIPS.

Table 2. Evaluation results to revise mappings

Mappings	Strategy	# of Unsatisf. Concepts	# of MUPS All	# of MUPS Avg	MUPS_Size Avg	# of Removed Axioms	Time seconds
CONFTOOL-CMT	Algorithm 1	26	351	14	6	4	331
	Algorithm 2	26	351	14	6	8	322
	Algorithm 3	4	15	4	5	8	12
EKAW-CMT	Algorithm 1	18	372	21	5	16	867
	Algorithm 2	18	372	21	5	15	863
	Algorithm 3	5	62	12	5	14	51
CRS-SIGKDD	Algorithm 1	19	64	3	5	5	18
	Algorithm 2	19	64	3	5	10	18
	Algorithm 3	5	13	3	4	7	6

does not need to handle all the unsatisfiable concepts, for example, 4 by Algorithm 3 versus 26 by other algorithms for *CONFTOOL-CMT*. Algorithm 1 has similar efficiency as Algorithm 2. This shows the efficiency to resolve incoherence using confidence values is similar to that using scoring function, since both algorithms share the same procedure to debug incoherence, but apply different strategies to resolve it.

Regarding to the effectiveness, Algorithm 2 removes more axioms than Algorithm 1 to restore the coherence in most cases. The reason is that for each found MIPS, there is always one axiom with the lowest confidence value. In such case, we have no other choice but removing this axiom when using confidence values to resolve incoherence. But for Algorithm 1, usually we have several choices for each MIPS. Therefore, by applying the Hitting Set Tree algorithm, Algorithm 1 can find a hitting set which is cardinality-smaller than that of Algorithm 2. For example, Algorithm 2 removes 8 axioms when repairing mappings in *CONFTOOL-CMT*, while Algorithm 1 removes only 4 axioms. But for *EKAW-CMT*, Algorithm 1 removes a few more axioms than Algorithm 2, because in most cases there are at least two axioms with the lowest confidence values for each MIPS. For all the test ontologies, Algorithm 3 removes less axioms than Algorithm 2. The reason is that Algorithm 3 may remove an axiom in a MUPS which belongs to several MIPS and Algorithm 2 always removes one axiom with the lowest confidence value in each MIPS.

To sum up, Algorithm 1 removes the least number of axioms in most cases, best complying with the requirement of minimal change. Algorithm 3 has excellent runtime performance compared with other two algorithms. At the same time, it sometimes removes fewer axioms than Algorithm 2. Thus it is the preferable option to deal with incoherence for large data sets when we have information about confidence values (or other ranking information) for axioms in the ontology.

Analysis of Meaningfulness. Table 3 shows the results for the meaningfulness of the repair based on the expert users' assessment whether the removal was correct. That is, if the definition of a removed axiom does not make sense according to the expert users' experience, we consider the removal as correct.

From Table 3 we can see that for all data sets and algorithms the rate of correct removals is considerable higher than that of the erroneous removals. This shows that

Table 3. Evaluation results for meaningfulness

Data set	Algorithm	# of Removed Axioms	Correctness	Error_Rate	Unknown_Rate
bt_km	Algorithm 1	27	0.41	0.28	0.31
	Algorithm 2	34	0.53	0.19	0.28
	Algorithm 3	33	0.65	0.13	0.22
CONFTOOL-CMT	Algorithm 1	4	0.56	0.31	0.13
	Algorithm 2	8	0.97	0.03	0
	Algorithm 3	8	0.97	0.03	0
EKAW-CMT	Algorithm 1	16	0.68	0.11	0.21
	Algorithm 2	15	0.64	0.05	0.31
	Algorithm 3	14	0.84	0.07	0.09
CRS-SIGKDD	Algorithm 1	5	0.60	0.40	0
	Algorithm 2	10	0.50	0.25	0.25
	Algorithm 3	7	0.79	0.07	0.14

generally that the ranking of axioms in our approach works well for resolving incoherence. The exact ratios largely depend on the data set. Especially the Unknown_Rate varies considerably for the different data sets; this is due to the nature of the data sets: For *bt_km*, there are many cases in which the users do not know whether the removal make sense or not, as the concepts in this data set are quite abstract like "model", "knowledge" and "order", it is hard to decide the relationships among those concepts. Comparing the meaningfulness results obtained by different algorithms, Algorithm 1 using scoring function is designed to comply with the principle of minimal change, and thus it typically removes fewer axioms. Yet, as it does not take any information about the confidence into account, Algorithm 2 and 3 using confidence values to resolve incoherence outperform the Algorithm 1 in terms of meaningfulness in most cases. For data set *CRS-SIGKDD*, the correctness for Algorithm 2 is higher than that for Algorithm 1, but the Error_Rate is much lower. Algorithm 3 consistently yields the most meaningful results. This shows that relying on confidence values, as provided by ontology learning tools applied to *bt_km*, or generated by ontology matching systems, leads to considerably more meaningful results when applying them for resolving incoherence.

7 Conclusions

In this paper, we have proposed a kernel revision operator for terminologies using an incision function. We have shown that our operator satisfies desirable logical properties. Further, we have provided two algorithms to instantiate our revision operator, one based on a scoring function and another one based on confidence values. Since these two algorithms need to compute all the MIPSs of the original ontology w.r.t. the new ontology, they are computationally very hard. Therefore, we have proposed an alternative algorithm which repairs the ontology by calculating MUPSs of the original ontology w.r.t. the new ontology and an unsatisfiable concept. According to our experimental results with real life ontologies, this last algorithm shows good scalability, although it may potentially remove slightly more axioms than the first one. An interesting future work is to explore efficient algorithms for generating minimal (or cardinality minimal) incision functions.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. J. Symb. Log. 50(2), 510–530 (1985)
2. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic EL+. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 52–67. Springer, Heidelberg (2007)
3. Castano, S., Ferrara, A., Montanelli, S.: Matching ontologies in open networked systems: Techniques and applications. pp. 25–63 (2006)
4. Falappa, M.A., Fermé, E.L., Kern-Isberner, G.: On the logic of theory change: Relations between incision and selection functions. In: Proc. of ECAI 2006, pp. 402–406 (2006)
5. Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: Inconsistencies, negations and changes in ontologies. In: Proc. of AAAI 2006, pp. 1295–1300 (2006)
6. Flouris, G., Plexousakis, D., Antoniou, G.: On applying the AGM theory to DLs and OWL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 216–231. Springer, Heidelberg (2005)
7. Haase, P., Stojanovic, L.: Consistent evolution of OWL ontologies. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 182–197. Springer, Heidelberg (2005)
8. Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., Sure, Y.: A framework for handling inconsistency in changing ontologies. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 353–367. Springer, Heidelberg (2005)
9. Haase, P., Völker, J.: Ontology learning and reasoning - dealing with uncertainty and inconsistency. In: Proc. of URSW 2005, pp. 45–55 (2005)
10. Hansson, S.O.: Kernel contraction. Journal of Symbolic Logic 59(3), 845–859 (1994)
11. Hansson, S.O.: A Textbook of Belief Dynamics: Theory Change and Database Updating. Kluwer Academic Publishers, Dordrecht (1999)
12. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
13. Katsuno, H., Mendelzon, A.O.: Propositional knowledge base revision and minimal change. Artificial Intelligence 52(3), 263–294 (1992)
14. Meilicke, C., Stuckenschmidt, H.: Applying logical constraints to ontology matching. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 99–113. Springer, Heidelberg (2007)
15. Qi, G., Hunter, A.: Measuring incoherence in description logic-based ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 381–394. Springer, Heidelberg (2007)
16. Qi, G., Liu, W., Bell, D.A.: Knowledge base revision in description logics. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 386–398. Springer, Heidelberg (2006)
17. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32(1), 57–95 (1987)
18. Ribeiro, M.M., Wassermann, R.: Base revision in description logics - preliminary results. In: Proc. of IWOD 2007, pp. 69–82 (2007)
19. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proc. of IJCAI 2003, pp. 355–362 (2003)
20. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. Journal of Automated Reasoning 39(3), 317–349 (2007)
21. Wassermann, R.: An algorithm for belief revision. In: Proc. of KR 2000, pp. 345–352 (2000)