ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-471

Conflict Resolution in Partially Ordered OWL DL Ontologies

Qiu Ji, Zhiqiang Gao[1] and **Zhisheng Huang**[2]

Abstract. Inconsistency handling in OWL DL ontologies is an important problem because an ontology can easily be inconsistent when it is generated or modified. Current approaches to dealing with inconsistent ontologies often assume that there exists a total order over axioms and use such an order to select axioms to remove. However, in some cases, such as ontology merging, a total order may not be available and we only have a partial order over axioms. In this paper, we consider a general notion of logical inconsistency and define the notion of *conflict* of an inconsistent ontology. We then propose a general approach to resolving inconsistency of a partially ordered ontology. We instantiate this approach by proposing two algorithms to calculate *prioritized hitting sets* for a set of conflicts. We implement the algorithms and provide evaluation results on the efficiency and effectiveness by considering both artificial and real-life data sets.

1 Introduction

Ontologies play a prominent role in formal representation of knowledge on the Semantic Web and the Web Ontology Language (OWL) has been standardized by W3C as an ontology language. One of the advantages of employing OWL in knowledge engineering is that reasoning services can be exploited to derive implicit knowledge from explicit knowledge stated in an OWL ontology. However, it becomes useless when an OWL ontology is inconsistent (i.e., there is no model for the ontology). Inconsistencies frequently occur within the ontology lifecycle, such as ontology learning [5] and ontology change [14]. Thus, handling inconsistencies in OWL ontologies is an important problem. In this paper, we focus on dealing with OWL DL ontologies since description logics provide a well-defined formal semantics for OWL DL which is a key sublanguage of OWL.

When dealing with inconsistencies in OWL DL ontologies, one can utilize priority information to select desirable repair plan(s). Typical examples of priority information are a trust order over axioms of an ontology [18] and the certainty degrees attached to the axioms [7]. Although there are many approaches to repairing an inconsistent OWL DL ontology (e.g., [17] and [11]), very few of them take into account the priority information. Furthermore, the approaches that explore priority information to deal with inconsistencies, such as [14] and [6], are often based on a total order over the axioms of an inconsistent ontology [7]. However, in some cases, such as ontology merging, a total order may not be available and we only have a partial order. These approaches cannot be applied to deal with partially ordered inconsistent ontologies. A partially ordered inconsistent ontology can be obtained in many cases. For example, when merging or integrating multiple ontologies through their mappings, we may get a partially ordered ontology that is inconsistent (see Section 4.2).

In this paper, we consider a general notion of logical inconsistency given in [17], which captures two common kinds of logical inconsistency in description logics, i.e., inconsistency and incoherence. Based on this notion, we define the notion of *conflict* of an inconsistent ontology. To resolve the conflicts of a partially ordered OWL DL ontology, we propose a general approach inspired by the general schemata of handling *locally stratified conflicts* in propositional logic given in [3]. We instantiate the approach by proposing two algorithms to calculate *prioritized hitting sets* for a set of conflicts. The notion of a prioritized hitting set generalizes the notion of a hitting set in [16] by taking into account of the partial order. The found prioritized hitting sets will be used to resolve conflicts. We implement the proposed algorithms and evaluate them with different settings according to the efficiency and effectiveness based on an artificial data set and a real-life data set for revealing their pros and cons.

2 Preliminary

We assume the readers are familiar with description logics (DLs) and refer them to [1] for more details. Although our approach to dealing with inconsistencies of an ontology is applicable to first-order logic, it is motivated by dealing with OWL DL ontologies with DL semantics. Thus, we use DL terminologies throughout the paper. In DLs, an important problem is to deal with logical inconsistencies which consist of two common kinds: inconsistency and incoherence. An ontology is *inconsistent* if it has no model and an ontology is *incoherent* if it contains at least one *unsatisfiable* concept which is interpreted as an empty set.

In the following, we introduce some definitions which are useful to explain logical inconsistencies.

Definition 1 *[19] Given an ontology O and an unsatisfiable concept C in O, a set $O' \subseteq O$ is a* minimal unsatisfiability-preserving subontology (MUPS) *of O w.r.t. C if C is unsatisfiable in O' and satisfiable in every subontology $O'' \subset O'$.*

A MUPS of O w.r.t. C is a minimal subontology of O in which C is unsatisfiable.

Definition 2 *[19] Let O be an incoherent ontology. An ontology $O' \subseteq O$ is a* minimal incoherence-preserving subontology (MIPS) *of O if O' is incoherent and every subontology $O'' \subset O'$ is coherent.*

[1] School of Computer Science and Engineering, Southeast University; Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China, email: jiqiu,zqgao@seu.edu.cn
[2] Department of Mathematics and Computer Science, Vrije University Amsterdam, The Netherlands, email: huang@cs.vu.nl

A MIPS of O is a minimal subontology of O which is incoherent. Note that a MIPS must be a MUPS, but not vice versa.

Definition 3 *[7] An ontology $O' \subseteq O$ is a minimal inconsistent subontology (MIS) of O, if O' is inconsistent and every subontology $O'' \subset O'$ is consistent.*

That is, by removing one axiom from each MIS we will get a consistent subontology.

Practical algorithms have been given to calculate MUPS, MIPS and MIS of a given ontology (see [19] and [7]).

In [17], the authors propose a general notion of an inconsistent ontology, which will be used in our work. Given a set U of unwanted axioms, an ontology O is a generalized inconsistent (abbreviated as *g-inconsistent*) ontology w.r.t. U if it infers some of the axioms in U. Otherwise, O is generalized consistent (abbreviated as *g-consistent*). It needs to be mentioned that, an unwanted axiom could be any axiom inferred by O and none of the axioms in U are equivalent to the tautology $\bot \sqsubseteq \top$. Based on this notion, we define the notion of a *conflict* of an inconsistent ontology.

Definition 4 *Given a set U of unwanted axioms and an ontology O, $M \subseteq O$ is a* conflict *w.r.t. U if it satisfies the conditions:*

- *M is a g-inconsistent ontology w.r.t. U,*
- *every $M' \subset M$ is a g-consistent ontology w.r.t. U.*

That is, a conflict of an ontology O is a minimal subontology of O which is g-inconsistent w.r.t. U. Clearly, if U is taken as $\{C \sqsubseteq \bot\}$, $\{C \sqsubseteq \bot : \text{C is a named concept}\}$ or $\{\top \sqsubseteq \bot\}$, then a conflict of O is a MUPS, MIPS and MIS respectively.

3 Conflict Resolution in Partially Ordered OWL DL Ontologies

In this paper, we consider an inconsistent OWL DL ontology $O = O_S \cup O_T$. O_S is assumed to be coherent and the axioms in it are surely to be true or reliable and cannot be ignored. The axioms in O_T are subject to change. Note that if $O_S = \emptyset$, then any axiom in O is subject to change. A partially ordered OWL DL ontology is a pair $(O, \preceq)$, where $\preceq$ is a partial order over axioms in O which satisfies the following condition: for any axiom $\phi \in O_S$, we have $\psi \prec \phi$ for all $\psi \in O_T$, where $\psi \prec \phi$ denotes $\psi \preceq \phi$ but $\phi \not\preceq \psi$. That is, any axiom in O_S should be preferred to all axioms in O_T. When $\psi \prec \phi$, we say that ϕ is preferred to ψ w.r.t. $\preceq$. In addition, $\psi \simeq \phi$ is used to denote $\psi \preceq \phi$ and $\phi \preceq \psi$, and indicates ψ is equal to ϕ w.r.t. $\preceq$.

Our definition of a partially ordered OWL DL ontology O is general to capture many applications, such as the trust-based revision [6] and reasoning in the presence of access restrictions [2]. When the partial order $\preceq$ is total, i.e., any two axioms in O are comparable, then O can be considered as a stratified ontology $O = (O_1, O_2, ..., O_n)$, where the axioms in O_i are equal w.r.t. $\preceq$ but are preferred to the axioms in O_j with $j > i$.

The problem that we will deal with is informally described as follows: suppose O is a g-inconsistent ontology w.r.t. a set of unwanted axioms U, we will remove some axioms in O to render a (or several) g-consistent one(s) w.r.t. U with the help of the partial order $\preceq$.

In the following, we first introduce how to stratify a conflict by considering a partial order. For resolving such kind of conflicts, we define the notion of a prioritized hitting set. Based on these definitions, we propose our general approach for resolving conflicts and then instantiate it by giving two concrete algorithms.

3.1 Stratification of a conflict

Given a partially ordered OWL DL ontology $(O, \preceq)$, where O is a g-inconsistent ontology w.r.t. a set U of unwanted axioms, we resolve inconsistency by removing axioms of O from its conflicts by considering $\preceq$. Inspired from the work on local stratification of a conflict in propositional logic given in [3], we provide a stratification of a conflict of O w.r.t. U (see Definition 5). The intuition behind the stratification is that we partition a conflict M into two parts $\overline{M}$ and $\underline{M}$ and give priority to axioms in $\overline{M}$ when resolving the conflict.

Definition 5 *Given a partially ordered OWL DL ontology $(O, \preceq)$, where $O = O_S \cup O_T$ is a g-inconsistent ontology w.r.t. a set of unwanted axioms U, suppose M is a conflict of O w.r.t. U. A simple stratification of M is a partition $(\overline{M}, \underline{M})$, where $\underline{M} = \{\phi \in M \cap O_T : \nexists \psi \in M,\ \psi \prec \phi\}$ and $\overline{M} = M \setminus \underline{M}$.*

That is, $\underline{M}$ consists of the axioms that belong to O_T and are least prioritized w.r.t. $\preceq$. We call $\underline{M}$ and $\overline{M}$ the lower stratum and upper stratum of M respectively.

It is easy to check that the following two properties hold for a simple stratification: (1) $O_S \cap \underline{M} = \emptyset$, and (2) $\underline{M} \neq \emptyset$.

3.2 Prioritized hitting sets

In [16], Reiter proposed the notion of a hitting set to calculate diagnosis for dealing with conflicts in a diagnostic reasoning system. Hitting sets have been used to resolve incoherence [19] and inconsistency [4] in DLs. Given a set of conflicts in a g-inconsistent ontology O w.r.t. a set U of unwanted axioms, a prioritized hitting set is a subset S of O such that S has overlap with each conflict. In this way, removing all axioms in S can resolve the inconsistency of O w.r.t. U. When resolving inconsistency in a partially ordered ontology, we may want to remove only those axioms appearing in the lower stratum of a conflict. But it is not always possible (see Example 1).

Example 1 *Assume an incoherent ontology O with $O_S = \emptyset$ (i.e., O is equal to O_T) contains the axioms:*

$\phi_1 : A \sqsubseteq B$, $\phi_2 : B \sqsubseteq C$, $\phi_3 : A \sqsubseteq \neg C$,
$\phi_4 : B \sqsubseteq \neg C$, $\phi_5 : \neg C \sqsubseteq \neg E$, $\phi_6 : B \sqsubseteq E$.

and has a partial order $\preceq$ such that $\phi_2 \prec \phi_4$, $\phi_3 \prec \phi_2$, $\phi_5 \prec \phi_4$ and $\phi_6 \prec \phi_4$. There are three MIPS of O: $M_1 = \{\phi_1, \phi_2, \phi_3\}$, $M_2 = \{\phi_2, \phi_4\}$ and $M_3 = \{\phi_4, \phi_5, \phi_6\}$. The simple stratification of M_1 is $(\underline{M_1}, \overline{M_1})$, where $\underline{M_1} = \{\phi_1, \phi_3\}$ and $\overline{M_1} = \{\phi_2\}$, the simple stratification of M_2 is $(\underline{M_2}, \overline{M_2})$, where $\underline{M_2} = \{\phi_2\}$ and $\overline{M_2} = \{\phi_4\}$, and the simple stratification of M_3 is $(\underline{M_3}, \overline{M_3})$, where $\underline{M_3} = \{\phi_5, \phi_6\}$ and $\overline{M_3} = \{\phi_4\}$. So ϕ_2 appearing in the lower stratum of M_2 but it is in the upper stratum of M_1.

In the following, we define the notion of a prioritized hitting set for a partially ordered OWL DL ontology.

Definition 6 *Given a partially ordered OWL DL ontology $(O, \preceq)$, where $O = O_S \cup O_T$ is a g-inconsistent ontology w.r.t. a set of unwanted axioms U, suppose $\mathcal{M}$ is a set of conflicts of O w.r.t. U. A prioritized hitting set H for $\mathcal{M}$ is a subset of O that satisfies the following conditions:*

1. *$H \cap M \neq \emptyset$ for each $M \in \mathcal{M}$,*
2. *$H \cap O_S = \emptyset$,*
3. *if $\phi \notin \cup_{M \in \mathcal{M}} \underline{M}$ then $\phi \notin H$.*

In Definition 6, condition 1 states that a prioritized hitting set should contain at least one axiom in every conflict. Condition 2 says that

a prioritized hitting set should not contain any axiom in O_S, and condition 3 says that if an axiom is not in the lower stratum of any conflict, then it should not be contained in a prioritized hitting set.

To resolve logical inconsistencies, the minimal change principle is often applied, which says we should remove as little information as possible. However, removing a prioritized hitting set from a g-inconsistent ontology may remove more axioms than necessary. Assume we have an incoherent ontology $O = \{\phi_1 : A \sqsubseteq B, \phi_2 : B \sqsubseteq C, \phi_3 : A \sqsubseteq \neg C, \phi_4 : B \sqsubseteq \neg C\}$, and a partial order $\preceq$ such that $\phi_1 \prec \phi_2$ and $\phi_4 \prec \phi_2$. There are two MIPS of O: $M_1 = \{\phi_1, \phi_2, \phi_3\}$ and $M_2 = \{\phi_2, \phi_4\}$. Then one prioritized hitting set for $\{M_1, M_2\}$ includes ϕ_1, ϕ_3 and ϕ_4. However, if we want to enforce the minimal change principle, we should remove ϕ_1 and ϕ_4. In order to fulfil both the minimal change principle and the preference principle, ideally, we should select prioritized hitting sets according to some minimal change strategy. However, this may be not practical because the computation of a minimal prioritized hitting set may be time-consuming. Thus, we consider some heuristics to compute prioritized hitting sets that are close to minimal ones.

3.3 Our general approach for resolving conflicts

To resolve conflicts of a partially ordered OWL DL ontology, we present a general approach inspired from the approach for handling locally stratified conflicts in propositional logic given in [3]. According to Definition 6, an axiom is removed to resolve inconsistency only if it belongs to the lower stratum of a conflict. In the following, we define the relatedness of two conflicts and use it to partition the set of conflicts. The intuition behind the relatedness and the partition is that conflict in different partitions are not related, thus we can deal with each partition one by one.

Definition 7 *Given two conflicts M_1 and M_2 of an ontology O, they are related if there exists an axiom ϕ in O_T such that $\phi \in M_1 \cap M_2$ and $\phi \in \underline{M_1} \cup \underline{M_2}$.*

Namely, if every axiom in O_T belonging to $M_1 \cap M_2$ is in neither of $\underline{M_1}$ nor $\underline{M_2}$, then M_1 and M_2 are unrelated in the sense that the resolution of M_i is not dependent of the resolution of M_j for $i \neq j$ and $i, j \in \{1, 2\}$. $M_1 \sim M_2$ is used to denote that M_1 and M_2 are related. Obviously, the relatedness relation is reflexive and symmetric, but is not transitive. If $M_1 \sim M_2$ and $M_2 \sim M_3$, M_1 and M_3 are indirectly related. We use $\sim^*$ to denote the transitive closure of $\sim$. Since $\sim^*$ is an equivalence relation, it can be used to partition a set $\mathcal{M}$ of conflicts. Each equivalence class consists of those conflicts that are (indirectly) related. When resolving conflicts, we can separately handle conflicts in each equivalence class induced by $\sim^*$. In Example 1, there are two equivalence classes induced by $\sim^*$, i.e., $\{M_1, M_2\}$ and $\{M_3\}$.

We can further define an ordering between conflicts in an equivalence class. Given two conflicts M_1 and M_2, M_1 has a positive influence on M_2 if the resolution of M_1 has an influence on the resolution of M_2. Thus, the resolution of M_1 should be done before that of M_2. We adapt the *positive influence* ordering given in [3] as follows.

Definition 8 *A conflict M_1 has a positive influence on a conflict M_2, denoted $M_1 \preceq_I M_2$, if and only if*

1. $\underline{M_1} \subseteq \underline{M_2}$, *or*
2. $\underline{M_1} \subseteq \underline{M_2}$ *and* $\underline{M_2} \not\subseteq \underline{M_1}$, *or*

We use $M_1 \prec_I M_2$ to denote $M_1 \preceq_I M_2$ but not $M_2 \preceq_I M_1$.

Algorithm 1: Repair partially ordered ontologies

Data: a partially ordered ontology $(O, \preceq)$, where $O = O_S \cup O_T$; a set of unwanted axioms U
Result: a set of repaired ontologies

1 **begin**
2 Let $\mathcal{M}_{\mathsf{U}}$ be a set of conflicts of O w.r.t. U;
3 Let $P = \{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_k\}$ be the partition of $\mathcal{M}_{\mathsf{U}}$;
4 **foreach** $\mathcal{M}_i$ *in* P **do**
5 Let $H_i = \emptyset$;
6 Let $C_i = \{(H_i, \mathcal{M}_i)\}$;
7 **while** $\exists\, (H, \mathcal{M}) \in C_i$, $\mathcal{M} \neq \emptyset$ **do**
8 $\mathcal{H} = \mathsf{HST}(min(\mathcal{M}))$;
9 $C_i = C_i \cup \{(H \cup H_{\mathsf{new}}, \mathcal{M} \setminus \{M : M \cap H_{\mathsf{new}} \neq \emptyset\}) : M \in \mathcal{M}, H_{\mathsf{new}} \in \mathcal{H}\} \setminus \{(H, \mathcal{M})\}$;
10 $\mathcal{H} = \{H_1 \cup H_2 \cup ... \cup H_k : (H_i, \mathcal{M}_i) \in C_i\}$;
11 $\mathcal{O} = \{O \setminus H : H \in \mathcal{H}\}$;
12 **return** $\mathcal{O}$

In Definition 8, if M_1 and M_2 satisfy either condition 1 or condition 2, then the resolution of M_1 necessarily results in the resolution of M_2. Consider Example 1 again, it is clear that $M_2 \preceq_I M_1$ but we do not have $M_1 \preceq_I M_2$, thus $M_2 \prec_I M_1$.

Since the ordering $\preceq_I$ is not transitive, we use $\preceq_I^*$ to denote the transitive closure of $\preceq_I$. If $M_1 \preceq_I^* M_2$ but $M_1 \not\preceq_I M_2$ and $M_2 \not\preceq_I M_1$, then we say that M_1 has an indirect positive influence on M_2. Given a set $\mathcal{M}$ of conflicts of O w.r.t. U, we use $min(\mathcal{M})$ to denote the set of conflicts that should be first resolved. The conflicts in $min(\mathcal{M})$ are minimal w.r.t. $\preceq_I^*$.

Based on the general schemata of handling locally stratified conflicts given in [3], we design an algorithm (see Algorithm 1) to resolve conflicts of a partially ordered ontology.

In Algorithm 1, we first partition the given set $\mathcal{M}_U$ of conflicts by using the transitive closure of relatedness relation $\sim$ (line 3). For each partition $\mathcal{M}_i$, we aim to generate a set of prioritized hitting sets for it. We use C_i to denote the set of all pairs of an incomplete prioritized hitting set and a set of conflicts to be resolved. For any such a pair with a non-empty set of conflicts, we first find the set of conflicts that are minimal w.r.t. $\preceq_I^*$ and then calculate a set of prioritized hitting sets by using a subroutine HST (line 8). After that, we update C_i (line 9). The specific subroutine will be given in the following subsection. Any such a subroutine will result in a specific algorithm for repairing partially ordered ontologies. After a set of prioritized hitting sets for each $\mathcal{M}_i$ is obtained, we use them to obtain a set of prioritized hitting sets for $\mathcal{M}_U$ (line 10). Finally, we get a set of repaired ontologies by removing from O axioms in a prioritized hitting set (line 11).

3.4 Algorithms to compute prioritized hitting sets

We present two algorithms for calculating prioritized hitting set(s) for a set of conflicts. Any of the two algorithms can be used as a subroutine HST in Algorithm 1. Before introducing our algorithms, we generalize the notion of a scoring function defined in [15].

Definition 9 *Given a set $\mathcal{M}$ of conflicts of a partially ordered ontology O w.r.t. a set of unwanted axioms U, the scoring function for O w.r.t. $\mathcal{M}$ is a function $S_{\mathcal{M}}$ such that $S_{\mathcal{M}}(\phi) = |\{M : \phi \in M\}|$.*

Namely, the score of an axiom ϕ w.r.t. $\mathcal{M}$ is the number of conflicts in $\mathcal{M}$ that contain the axiom. This score can be considered as the impact of its removal on resolving conflicts.

Algorithm 2: Calculate multiple prioritized hitting sets with scoring function

Data: a set of conflicts $\mathcal{M}$ of a partially ordered ontology O w.r.t. U
Result: a set of prioritized hitting set
begin
  foreach M *in* $\mathcal{M}$ **do**
    $H_M = \{\phi \in \underline{M} : \text{for any } \psi \in \underline{M}, S_{\mathcal{M}}(\phi) \geq S_{\mathcal{M}}(\psi)\}$;
  $\mathcal{C} = \{H_M : M \in \mathcal{M}\}$;
  $\mathcal{H} = \mathsf{HST}_{\mathsf{Reiter}}(\mathcal{C})$;
  return $\mathcal{H}$

Algorithm 3: Calculate a single prioritized hitting set

Data: a set of conflicts $\mathcal{M}$ of a partially ordered ontology O w.r.t. U
Result: a set of prioritized hitting set
begin
  foreach M *in* $\mathcal{M}$ **do**
    $H_M = \{\phi \in \underline{M} : \text{for any } \psi \in \underline{M}, S_{\mathcal{M}}(\phi) \geq S_{\mathcal{M}}(\psi)\}$;
  $\mathcal{H} = \cup_{i=1}^{|\mathcal{M}|}\{\phi_i\}$ ($\phi_i \in H_M$ and $M \in \mathcal{M}$);
  return $\mathcal{H}$

Our first algorithm (Algorithm 2) is based on Reiter's Hitting Set Tree (HST) algorithm given in [16] which is denoted by $\mathsf{HST}_{\mathsf{Reiter}}$. In our algorithm, for each conflict M in $\mathcal{M}$, we obtain a subset H_M of it that consists of the axioms in its lower stratum with the highest score. These subsets are taken as the input of the function $\mathsf{HST}_{\mathsf{Reiter}}$. Clearly, each set in $\mathcal{H}$ returned by $\mathsf{HST}_{\mathsf{Reiter}}(\mathcal{C})$ is a prioritized hitting set of $\mathcal{M}$. This algorithm utilizes one heuristics to approximate minimal prioritized hitting sets.

Since exponential number of prioritized hitting sets may be returned in the worst case of Algorithm 2, we propose another algorithm (see Algorithm 3) to output a single prioritized hitting set for a set of conflicts. This algorithm first computes H_M for each M in $\mathcal{M}$, and then randomly chooses one axiom from H_M of M in $\mathcal{M}$ and returns the union of these axioms.

4 Evaluation

In this section, we first introduce our implementation and then describe our data sets and evaluation results.

4.1 Implementation

Our algorithms were implemented in Java against the OWL API version 3.4.3[3] and Pellet API 2.3.1[4]. All standard reasoning tasks were preformed by using Pellet. To calculate the conflicts of a partially ordered ontology w.r.t. unwanted axioms, we adapted the algorithm given in [10]. All experiments have been performed on a laptop with 2.4GHz Intel(R) Core(TM)2 Duo CPU and 6GB RAM using Windows 7. The maximum Java heap space is set to 3GB. A time limit of 10,000,000 milliseconds (i.e., nearly 3 hours) is imposed on computing prioritized hitting sets for a given set of conflicts or calculating the conflicts for a given set of unsatisfiable concepts. The results, data sets and our implementation can be downloaded online[5].

ID	Ontology	$\lvert O_S \rvert$	$\lvert O_T \rvert$	Expressivity
km	Subontology of km1500	0	10,000	$\mathcal{ALC}$
om1	cmt-WikiMatch-ekaw-CID-sig.	575	14	$\mathcal{SHIN}(\mathcal{D})$
om2	confof-RIM-iasted-XMapGen-sig.	666	30	$\mathcal{SHIN}(\mathcal{D})$
om3	con.-ODG-confof-OntoK2-ekaw	714	23	$\mathcal{SHIN}(\mathcal{D})$
om4	cmt-MapSSS-con.-ODG-ekaw	744	27	$\mathcal{SHIN}(\mathcal{D})$
om5	cmt-XMa-confof-SYN.-edas	1,161	33	$\mathcal{SHOIN}(\mathcal{D})$
om6	cmt-ODG-confof-RIM-ekaw	655	32	$\mathcal{SHIN}(\mathcal{D})$

Table 1. Data sets.

4.2 Data sets

We performed the evaluation on an artificial data set to better reflect the difference w.r.t. the efficiency of our algorithms with different settings, and a real-life data set to see the performance in a real case. Although an unwanted axiom could be any kind of axiom, we only consider the unsatisfiability of the concepts (i.e., the axioms in the form of $C \sqsubseteq \bot$, C is a concept) in our evaluation.

The artificial data set was constructed based on ontology km1500 which was developed by applying ontology learning techniques. This ontology comprises 12,656 axioms originally and each axiom is associated with a confidence value. In order to have more than one partition of the conflicts to be considered, two disjoint sets of axioms were extracted from ontology km1500 and each set includes 5,000 axioms. The union of the two sets comprises ontology km (see Table 1). To make ontology km be a partially ordered ontology, we partition its axioms into four disjoint sets of axioms according to their confidence values. The axioms in one set are assumed not to be comparable with those axioms in other sets.

The real-life data set was constructed based on the data set of conference track provided by OAEI 2013[6]. The data set in this track consists of a set of ontologies and the mappings generated by those ontology mapping systems participated in the contest. To construct a partially ordered ontology, we take the union of three individual ontologies as the reliable axioms and their mappings as the axioms to be changed by translating a mapping into DL axioms [13, 14]. Then a partial order can be established: (1) The axioms in a mapping could be compared with each other. (2) The axioms in different mappings cannot be compared with each other. Table 1 presents more details about the selected mapped ontologies and the mappings among them. The name of a merged ontology consists of the first mapped ontology, the first mapping system, the second mapped ontology, the second mapping system and the third mapped ontology. Through merging the three mapped ontologies and the mappings generated by the systems, a partially ordered ontology is developed.

4.3 Evaluation Results

We evaluate our algorithms to compute multiple prioritized hitting sets (i.e., Algorithm 1 with Algorithm 2) with different settings by considering the efficiency and effectiveness[7]. We use P, I and S to indicate our algorithm using the partition technique, using influence order and using a scoring function respectively. The meaningful combinations consist of PIS, PS, S, PI and P. Besides, our algorithm without using P, I or S is taken as a baseline.

[3] http://owlapi.sourceforge.net/
[4] http://clarkparsia.com/pellet/
[5] http://atur.aturstudio.com/homepage/qiuji/conflicts.zip
[6] http://oaei.ontologymatching.org/2013/
[7] We did not evaluate our algorithm to compute a single prioritized hitting set as it is a special case of the evaluated ones or compare with existing algorithms since they cannot be directly applied to deal with partially ordered ontologies.

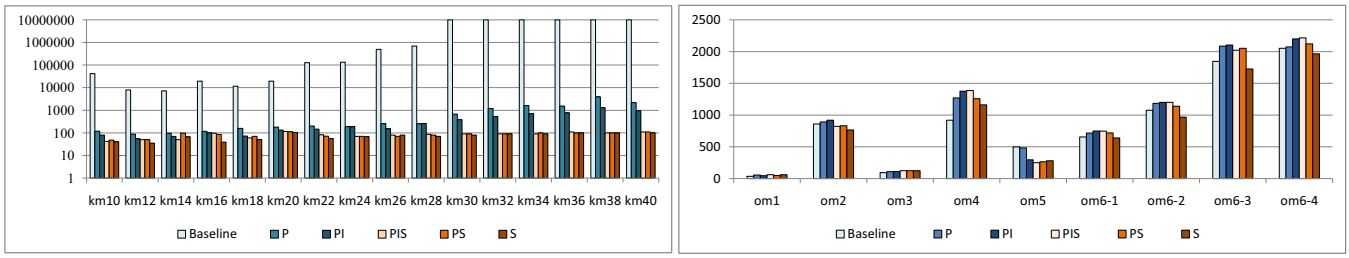

Figure 1. The time in milliseconds (Y axis) to compute prioritized hitting sets for a set of conflicts w.r.t. selected unsatisfiable concepts in an ontology.

ID	# UC		# Conflicts	Conflict Size		
	All	Selected		Min	Max	Mean
km	3,127	2	6,310	4	25	15.99
om1	18	18	8	6	9	6.75
om2	6	6	112	5	16	11.15
om3	20	20	21	4	12	8.62
om4	12	12	113	5	21	18.92
om5	64	64	29	3	20	12.21
om6-1	53	6	111	3	12	8.25
om6-2	53	5	172	3	12	7.99
om6-3	53	3	298	4	12	8.20
om6-4	53	4	344	4	12	7.93

Table 2. The conflicts w.r.t. selected unsatisfiable concepts.

Since the computation of conflicts is not one main focus of this paper, we only provide the details about the found conflicts in Table 2 without showing the time to compute conflicts. From the two disjoint sets of axioms in ontology km, we select an unsatisfiable concept from each set such that the two selected concepts are different. In this way, the conflicts w.r.t. the two concepts would have at least two partitions. Since quite a lot conflicts have been found for ontology km (i.e., more than 6,000 conflicts), we choose 16 subsets of conflicts with increasing sizes. The number of selected conflicts varies from 10 to 40 and we use "km-n" (n=10, 12, ..., 40) to indicate ontology km with n conflicts (see Figure 1).

4.3.1 *Efficiency Evaluation*

Figure 1 gives the evaluation results w.r.t. the efficiency of different algorithms for computing multiple prioritized hitting sets for a given set of conflicts.

From the results of ontology km given in Figure 1, we observe that the baseline algorithm is much more time-consuming than algorithms with other settings. On the one hand, partitioning conflicts improves the efficiency of computing multiple prioritized hitting sets. On the other hand, our algorithms using a scoring function outperform other algorithms. Both heuristics could largely reduce the search space of constructing a hitting set tree.

From Figure 1, we also see that all algorithms with different settings could finish the computation of prioritized hitting sets quickly (i.e., within 2500 milliseconds) for real-life ontologies, even if more than 300 conflicts are handled (see Table 2). This is because the lower stratum of each found conflict of such an ontology often has one or two axioms and most of the strata have overlap. In addition, the algorithms without computing the partitions or influence order slightly outperform others for most of these ontologies. It is because the time to compute a partition or an influence order cannot be ignored when little time has been spent on computing prioritized hitting sets.

4.3.2 *Effectiveness Evaluation*

To evaluate the effectiveness of our algorithms, we compare not only the number of found prioritized hitting sets but the effect of any two sets of prioritized hitting sets on resolving conflicts.

Ontology ID	# Prioritized hitting sets					
	Baseline	P	PI	PIS	PS	S
km10	8,004	8,004	1,344	12	12	12
km12	2,596	2,596	504	12	8	8
km14	3,237	4,233	756	6	6	6
km16	4,300	6,050	378	9	6	6
km18	3,225	5,063	378	9	6	6
km20	3,999	5,429	420	9	6	6
km22	8,268	12,212	800	2	6	6
km24	7,800	8,896	768	2	4	4
km26	12,166	20,700	864	2	2	2
km28	12,474	13,680	928	2	2	2
km30	-	50,416	4,440	2	2	2
km32	-	108,228	9,720	2	2	2
km34	-	137,982	9,720	2	2	2
km36	-	154,686	11,664	2	2	2
km38	-	324,485	17,136	4	2	2
km40	-	202,608	7,616	4	2	2
om1	2	2	2	1	1	1
om2	8	8	1	1	1	1
om3	2	2	2	1	1	1
om4	2	2	2	2	1	1
om5	64	64	27	2	1	1
om6-1	1	1	1	1	1	1
om6-2	1	1	1	1	1	1
om6-3	2	2	1	1	1	1
om6-4	2	2	1	1	1	1

Table 3. The number of found prioritized hitting sets.

Table 3 shows the number of found prioritized hitting sets, where '-' means an algorithm fails to finish its computation within limited time. We first observe that the scoring function-based algorithms find much less prioritized hitting sets than other algorithms. Besides, when comparing the effectiveness of the algorithms without using a scoring function, we obtain the following observations. (1) The partition-based algorithm always finds more prioritized hitting sets than others. This is due to the different ways to construct hitting set trees. The partition-based algorithm constructs a hitting set tree for each partition and then combines those hitting sets obtained from different partitions. But the baseline algorithm only needs to construct one hitting set tree over all considered conflicts. (2) Our algorithm with the setting PI finds much less prioritized hitting sets than others because computing an influence order has largely reduced the number of conflicts to be handled. For example, the conflicts in ontology km18 are partitioned into two sets, each of which contains 9 conflicts. By computing an influence order, we found 10 conflicts in total to be resolved first and finally obtained 378 prioritized hitting sets. In the case of not computing an influence order, more than 3,000 prioritized hitting sets have been computed.

The effect of any two sets of prioritized hitting sets on resolving conflicts can be measured in the following way. One set of prioritized hitting sets $\mathcal{H}_1$ is contained by the other set of prioritized hitting set $\mathcal{H}_2$ if each prioritized hitting set in $\mathcal{H}_1$ is contained by a prioritized hitting set in $\mathcal{H}_2$. $\mathcal{H}_1$ is equivalent to $\mathcal{H}_2$ w.r.t. the effect of resolving conflicts if $\mathcal{H}_1$ is contained by $\mathcal{H}_2$ and $\mathcal{H}_2$ is contained by $\mathcal{H}_1$. In our experiments, the available sets of prioritized hitting sets found by the three algorithms without using a scoring function are equivalent to each other for each given set of conflicts. This shows that the baseline algorithm and the algorithm with the setting P have found quite a lot

of redundant prioritized hitting sets by comparing with the algorithm with the setting PI. Among the scoring function-based algorithms, a set of prioritized hitting sets found by our algorithm with the setting PIS may not be equivalent to the corresponding set of prioritized hitting sets found by our algorithm with the setting PS. Because the scores of the axioms in a set of conflicts $\mathcal{M}$ are different from the scores of the axioms in the conflicts that should be first resolved (i.e., $min(\mathcal{M})$), which results in different search spaces.

5 Related Work

In [14], two algorithms were proposed to revise an ontology by considering the weights attached to axioms. Namely, they choose an arbitrary axiom with the lowest weights from each MIPS or MUPS to remove by applying a hitting set tree algorithm. These algorithms can be seen as a special case of our general algorithm. The authors also proposed an algorithm for revising an ontology using scoring function. However, this algorithm does not use any priority information and is a special case of our algorithm with the setting S by assuming all axioms in an ontology share the same priority.

In [6], the authors proposed a trust-based revision approach for expressive web syndication, where the Semantic Web languages are used to represent the content of publications and subscriptions. In such a scenario, any publisher could publish any statement or axiom and the inconsistency may occur when merging multiple statements. To resolve the inconsistency, a repair algorithm was designed to randomly select an axiom with the lowest trust value from a conflict to remove. This repair algorithm is a special case of our general algorithm because they assume a total order over axioms. Compared with our instantiated algorithm using Algorithm 3, this algorithm only utilizes the trust values, whilst our algorithm considers both ordering information and scores of axioms.

This work is also related to the work on mapping repair (see [9] for different mapping repair systems). The problem of mapping repair is to resolve incoherence when two ontologies are merged. However, all the algorithms for mapping repair, such as those given in [12] and [8], assume a total order over merged ontology, i.e., mapped ontologies are more important than mappings and each correspondence of a mapping is attached with a weight. Furthermore, mapping repair algorithms often consider merging of two ontologies, whilst our approach can deal with merging of more than two ontologies mapped by different mapping systems.

6 Conclusions and future work

In this paper, we proposed a general approach to resolving conflicts of a partially ordered ontology. Our approach is general in the following sense. First, it can be applied to deal with all kinds of conflicts of OWL DL ontologies, such as MUPS, MIPS and MIS. Second, our approach uses a subroutine to calculate one or more prioritized hitting sets for a set of conflicts. Third, the algorithms proposed in some existing work are special cases of ours (see Section 5).

We also implemented the algorithms and did experiments on the artificial and real-life data sets. The experimental results showed that partitioning conflicts could largely improve the efficiency of our algorithm. Besides, although computing influence order has not shown obvious improvement w.r.t. the efficiency, it helps a lot to reduce the number of redundant prioritized hitting sets. When applying a scoring function, our algorithms have shown very good efficiency, but only a few prioritized hitting sets could be found.

We have applied our approach to resolving conflicts caused by merging mapped ontologies. As a future work, we will apply our approach to resolving conflicts of partially ordered ontologies where the partial order is either obtained by trust levels of ontologies [6] or obtained by access right [2].

Acknowledgements

We gratefully acknowledge funding from the National Science Foundation of China under grants 61170165.

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, *The Description Logic Handbook: Theory, Implementation and Application*, Cambridge University Press, 2007.
[2] Franz Baader, Martin Knechtel, and Rafael Peñaloza, 'A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms', in *ISWC*, pp. 49–64, (2009).
[3] Salem Benferhat and Laurent Garcia, 'Handling locally stratified inconsistent knowledge bases', *Studia Logica*, **70**(1), 77–104, (2002).
[4] Jianfeng Du and Guilin Qi, 'Decomposition-based optimization for debugging of inconsistent OWL DL ontologies', in *KSEM*, pp. 88–100, (2010).
[5] Daniel Fleischhacker, Christian Meilicke, Johanna Völker, and Mathias Niepert, 'Computing incoherence explanations for learned ontologies', in *RR*, pp. 80–94, (2013).
[6] Jennifer Golbeck and Christian Halaschek-Wiener, 'Trust-based revision for expressive web syndication', *Journal of Logic and Computation*, **19**(5), 771–790, (2009).
[7] Peter Haase and Johanna Völker, 'Ontology learning and reasoning - dealing with uncertainty and inconsistency', in *URSW*, pp. 366–384, (2008).
[8] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks, 'Large-scale interactive ontology matching: Algorithms and implementation', in *ECAI*, pp. 444–449, (2012).
[9] Ernesto Jiménez-Ruiz, Christian Meilicke, Bernardo Cuenca Grau, and Ian Horrocks, 'Evaluating mapping repair systems with large biomedical ontologies', in *DL*, pp. 246–257, (2013).
[10] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin, 'Finding all justifications of OWL DL entailments', in *ISWC/ASWC*, pp. 267–280, (2007).
[11] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau, 'Repairing unsatisfiable concepts in OWL ontologies', in *ESWC*, pp. 170–184, (2006).
[12] Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin, 'Reasoning support for mapping revision', *Journal of Logic and Computation*, **19**(5), 807–829, (2009).
[13] Christian Meilicke, Johanna Völker, and Heiner Stuckenschmidt, 'Learning disjointness for debugging mappings between lightweight ontologies', in *EKAW*, pp. 93–108, (2008).
[14] Guilin Qi, Peter Haase, Zhisheng Huang, Qiu Ji, Jeff Z. Pan, and Johanna Völker, 'A kernel revision operator for terminologies - algorithms and evaluation', in *ISWC*, pp. 419–434, (2008).
[15] Guilin Qi and Anthony Hunter, 'Measuring incoherence in description logic-based ontologies', in *ISWC*, pp. 381–394, (2007).
[16] Raymond Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence*, **32**(1), 57–95, (1987).
[17] Márcio Moretto Ribeiro and Renata Wassermann, 'Base revision for ontology debugging', *Journal of Logic and Computation*, **19**(5), 721–743, (2009).
[18] Simon Schenk, 'On the semantics of trust and caching in the semantic web', in *ISWC*, pp. 533–549, (2008).
[19] Stefan Schlobach and Ronald Cornet, 'Non-standard reasoning services for the debugging of description logic terminologies', in *IJCAI*, 355–362, (2003).