Entailment-based Axiom Pinpointing in Debugging Incoherent Terminologies

Yuxin Ye[1,2], Dantong Ouyang[1,2⋆] , Jing Su[2]

[1] College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Changchun 130012, China
{yeyx,ouyd}@jlu.edu.cn
sujing11@mails.jlu.edu.cn

Abstract. One of the major problems of axiom pinpointing for incoherent terminologies is the precise positioning within the conflict axioms. In this paper we present a formal notion for the entailment-based axiom pinpointing of incoherent terminologies, where the parts of an axiom is defined by atomic entailment. Based on these concepts, we prove the one-to-many relationship between existing axiom pinpointing with the entailment-based axiom pinpointing. For its core task, calculating minimal unsatisfiable entailment, we provide algorithms for OWL DL terminologies using incremental strategy and Hitting Set Tree algorithm. The feasibility of our method is shown by case study and experiment evaluations.

Keywords: ontology debugging, description logics, pinpointing, MUPS

1 Introduction

Ontology debugging becomes a challenging task for ontology modelers since the improvement of expressivity of ontology language and ontology scale[1]. Axiom pinpointing [2] is an important mean for ontology debugging. Any approach which can detect a set of axioms in the terminology that lead to logic conflict is belong to axiom pinpointing. It can be categorized into MSSs (maximally satisfiable sub-TBox), MUPS (minimal unsatisfiable sub-TBox) and justification. For finding maximally concept-satisfiable terminologies, Meyer[3] proposes a tableau like procedure for terminologies represented in $\mathcal{ALC}$. The approach of Meyer is extended by Lam[4] to get a fine-grained axiom pinpointing for $\mathcal{ALC}$ terminologies. In addition, several methods have been proposed to calculate the MUPS. Schlobach and Cornet[5] provide complete algorithms for unfoldable $\mathcal{ALC}$-TBox based on minimization of axioms for MUPS, then Schlobach[6, 7] presents a framework for the debugging of logically contradicting terminologies. Parsia[8]

⋆ Corresponding author: Dantong Ouyang, Email: ouyd@jlu.edu.cn extends Schlobach[5] to more expressive DLs. Baader[2] presents automata-based algorithms for reasoning in DLs with the pinpointing formula whose minimal valuations correspond to the MUPS. From the perspective of unsatisfiability, justification is the MUPS of an unsatisfiable concept. Kalyanpur has explored the dependencies between unsatisfiable classes[9], and proposed several approaches for computing all justifications of an entailment in an OWL-DL ontology[10]. Debugging tasks in OWL ontologies are in general computationally hard, so some optimization techniques are introduced for ontology debugging such as heuristic method[11] of identifying common errors and inferences, and modularization[12] for large ontologies. On the whole, various approaches achieve the result sets of axioms responsible for an unsatisfiable concept or a incoherent terminology. Hasse[13] provides a set of criteria for comparing between different approaches related to ontology debugging directly or indirectly, and that none of the surveyed approaches is universally applicable for any application scenario. Axiom pinpointing identifies conflict axioms, but practical problems remain. It is not clear which parts of axioms lead to the conflicts. and some contradictions would be lost[14]. In this paper, we try to give other notion of axiom pinpointing for incoherent terminologies, and define algorithms for this task.

The rest of this paper is organized as follows. In section 2 we briefly introduce the drawback of MUPS. Then the formal definitions about fine-grained axiom pinpointing, and the link with axiom pinpointing are presented in section 3. Section 4 presents algorithm for calculating the minimal unsatisfiable entailment. Section 5 analyzes the fine-grained axiom pinpointing with a case study and evaluates the algorithm by experimenting with common ontologies. Finally, we conclude the paper in section 6.

2 Drawback of MUPS

Axiom pinpointing[2] has been introduced in description logics to help the user to understand the reasons why consequences hold and to remove unwanted consequences by computing minimal subsets of the terminology that have the consequence. The axiom pinpointing we discuss in this paper is MUPS[5]. It's useful for relating sets of axioms to the unsatisfiability of specific concept.

Definition 1 (MUPS[5]). *A TBox $\mathcal{T}' \subseteq \mathcal{T}$ is a minimal unsatisfiability preserving sub-TBox* (MUPS) *for C in $\mathcal{T}$ if C is unsatisfiable in $\mathcal{T}'$, and C is satisfiable in every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$. The set of all* MUPS *of C in $\mathcal{T}$ is denoted as $mups(\mathcal{T}, C)$.*

Most existing approaches can obtain the different fine-grained problematic axioms on the basis of axiom pinpointing as none of these approaches define exactly what they mean by parts of axioms. Further, some logic contradictions would be lost with axiom pinpointing since it does not point out the specific location within the axioms of the logic contradiction. Let us use an example to illustrate these limitations.

Example 1. A TBox $\mathcal{T}_1$ consists of the following axioms ($\alpha_1 - \alpha_6$), where A and B are base concepts, $A_1, ..., A_6$ are named concepts, and r and s are roles:

$\alpha_1 : A_1 \sqsubseteq A_2 \sqcap \exists r.A_2 \sqcap A_3$	$\alpha_4 : A_4 \sqsubseteq \forall r.A \sqcap \forall s.B \sqcap A_5$
$\alpha_2 : A_2 \sqsubseteq A \sqcap B$	$\alpha_5 : A_5 \sqsubseteq \exists r.\neg A \sqcap A_6$
$\alpha_3 : A_3 \sqsubseteq \forall r.(\neg B \sqcap \neg A)$	$\alpha_6 : A_6 \sqsubseteq \exists s.\neg B$

Consider the above example, by using standard DL TBox reasoning, it can be shown that the concept A_1 and A_4 are unsatisfiable. Analyzing concept A_1, the existing approaches identify $\{\alpha_1, \alpha_2, \alpha_3\}$ the only MUPS for A_1 in $\mathcal{T}_1$, but it is not clear whether A_2 or $\exists r.A_2$ of α_1 contradicts with α_3. In addition, it hides crucial information, e.g., that unsatisfiability of A_1 depends on all parts of α_2 or α_3. For A_4, $\{\alpha_4, \alpha_5\}$ is the only MUPS for A_4 in $\mathcal{T}_1$, which on behalf of the error caused by $\forall r.A$ of α_4 and $\exists r.\neg A$ of α_5. Actually, $\forall s.B$ of α_4, A_6 of α_5 and $\forall s.\neg B$ of α_6 also lead to the unsatisfiability of A_4, which would not be involved in the reason of unsatisfiability of A_4 since MUPS can not pinpoint the location of conflict, i.e., some unsatisfiable (or incoherent) reasons would be ignored by axiom pinpointing. We will use this example to explain our debugging methods.

3 Entailment and MUE

This section presents the main technical contribution of the paper. We would like to provide a framework for entailment-based axiom pinpointing. We will present the formal definitions which involve MUE, then show the relationship between MUPS and MUE. The second subsection is concerned with how to get all components w.r.t. axiom and terminology.

3.1 Formal Definitions

To compensate the limitations of axiom pinpointing, we introduce the notion of fine-grained axiom pinpointing and link it to description logic-based systems. Whereas the definitions of fine-grained axiom pinpointing are independent of the choice of a particular Description Logic.

Definition 2 (Entailment[17]). *Given a logical language $\mathcal{L}$, an entailment $\models$ states a relation between an terminology $\mathcal{T}$ and an axiom $\alpha \in \mathcal{L}$. We use $\mathcal{T} \models \alpha$ to denote that the ontology $\mathcal{T}$ entails the axiom α. Alternatively, we say that α is a consequence of the terminology $\mathcal{T}$ under entailment relation $\models$. The entailment relation is said to be a standard one if and only if α is always holds in any model in which the terminology $\mathcal{T}$ holds, i.e., for any model $\mathcal{I}$, $\mathcal{I} \models \mathcal{T} \Rightarrow \mathcal{I} \models \alpha$.*

Definition 3 (Atomic Entailment). *Let $\mathcal{T}$ be a terminology and β be an axiom such that $\mathcal{T} \models \beta$. We call $\models$ is an atomic entailment between $\mathcal{T}$ and β if $\{\beta\}$ has no consequence but β. Alternatively, β is an atomic consequence of $\mathcal{T}$.*

We denote by $\mathcal{E}(\mathcal{T})$ and $\mathcal{E}(\alpha)$ the set of all atomic consequences of terminology $\mathcal{T}$ and $\{\alpha\}$, respectively. If a terminology $\mathcal{T}$ is incoherent, then for any axiom

β, $\mathcal{T} \models \beta$, i.e., a standard entailment is explosive. Thus, we require $\mathcal{E}(\mathcal{T}) = \bigcup_{\alpha \in \mathcal{T}} \mathcal{E}(\alpha)$ if $\mathcal{T}$ is incoherent, and $\mathcal{E}(\alpha) = \{\alpha\}$ if the axiom α has no model. Intuitively, an atomic consequence of an axiom is a part of the axiom, and set of all atomic consequences of an axiom contains all parts of the axiom.

Definition 4 (MUE). *Let C be an unsatisfiable concept in terminology $\mathcal{T}$. A sub-TBox $\mathcal{T}_c \subseteq \mathcal{E}(\mathcal{T})$ is a minimal unsatisfiable entailment for C in $\mathcal{T}$ if C is unsatisfiable in $\mathcal{T}_c$, and C is satisfiable in every sub-TBox $\mathcal{T}'_c \subset \mathcal{T}_c$.*

The entailment-based axiom pinpointing inferential service is the problem of computing MUE. We denote by $mue(\mathcal{T}, C)$ the set of MUE of C in terminology $\mathcal{T}$. In the terminology of Reiter's diagnosis each $mue(\mathcal{T}, C)$ is a collection of conflict sets. The following are the MUE for our example TBox $\mathcal{T}_1$:

$$\begin{aligned} mue(\mathcal{T}_1, A_1) = \{&\{A_1 \sqsubseteq \exists r.A_2, A_1 \sqsubseteq A_3, A_2 \sqsubseteq A, A_3 \sqsubseteq \forall r.\neg A\},\\ &\{A_1 \sqsubseteq \exists r.A_2, A_1 \sqsubseteq A_3, A_2 \sqsubseteq B, A_3 \sqsubseteq \forall r.\neg B\}\}\\ mue(\mathcal{T}_1, A_4) = \{&\{A_4 \sqsubseteq \forall r.A, A_4 \sqsubseteq A_5, A_5 \sqsubseteq \exists r.\neg A\},\\ &\{A_4 \sqsubseteq \forall s.B, A_4 \sqsubseteq A_5, A_5 \sqsubseteq A_6, A_6 \sqsubseteq \exists s.\neg B\}\} \end{aligned}$$

MUE can be regarded as the fine-grained axiom pinpointing for MUPS. The relationship between axiom pinpointing and our pinpointing is established by Theorem 1, i.e., the one-to-many relationship between MUPS and MUE.

Theorem 1 (MUPS-to-MUEs relationship). *Let C be an unsatisfiable concept in terminology $\mathcal{T}$. Then:*
(1) If C is unsatisfiable in $\mathcal{T}$, then C is unsatisfiable in $\mathcal{E}(\mathcal{T})$.
(2) for every $\mathcal{M} \in mups(\mathcal{T}, C)$, there is a $\mathcal{K} \in mue(\mathcal{T}, C)$ s.t. $\mathcal{K} \subseteq \mathcal{E}(\mathcal{M})$.
(3) for any $\mathcal{M}_1, \mathcal{M}_2 \in mups(\mathcal{T}, C)$ and $\mathcal{K}_1, \mathcal{K}_2 \in mue(\mathcal{T}, C)$ where $\mathcal{M}_1 \neq \mathcal{M}_2$, $\mathcal{K}_1 \subseteq \mathcal{E}(\mathcal{M}_1)$ and $\mathcal{K}_2 \subseteq \mathcal{E}(\mathcal{M}_2)$, we have $\mathcal{K}_1 \neq \mathcal{K}_2$.

Proof. We prove (1), (2) and (3) in order.
(1) According to the definition of atomic entailment, If an axiom $\alpha \in \mathcal{T}$ has no model, $\mathcal{E}(\alpha)$=$\{\alpha\}$. Otherwise, we can prove $\{\alpha\}$ and $\mathcal{E}(\alpha)$ are equivalent with the axiom decomposition which is described in next subsection. In general, $\mathcal{T}$ and $\mathcal{E}(\mathcal{T})$ are equivalent.
(2) Since $\mathcal{M} \subseteq mups(\mathcal{T}, C)$, we have C is unsatisfiable in $\mathcal{E}(\mathcal{M})$. Thus, $mue(\mathcal{M}, C) = mups(\mathcal{E}(\mathcal{M}), C)$, and for every $\mathcal{K} \in mue(\mathcal{M}, C)$, we get $\mathcal{K} \subseteq \mathcal{E}(\mathcal{M})$.
(3) Suppose $\mathcal{K} \in mue(\mathcal{T}, C), \mathcal{M}_1, \mathcal{M}_2 \in mups(\mathcal{T}, C)$ $(\mathcal{M}_1 \neq \mathcal{M}_2)$ s.t. $\mathcal{K} \subseteq \mathcal{E}(\mathcal{M}_1)$ and $\mathcal{K} \subseteq \mathcal{E}(\mathcal{M}_2)$. Thus, there exists a sub-TBox $\mathcal{T}' \subseteq \mathcal{T}$ s.t. $\mathcal{K} \subseteq \mathcal{E}(\mathcal{T}')$ and $\mathcal{K} \nsubseteq \mathcal{E}(\mathcal{T}'')$ for every $\mathcal{T}'' \subset \mathcal{T}'$. Then C is unsatisfiable in $\mathcal{T}'$, $\mathcal{T}' \subseteq \mathcal{M}_1$ and $\mathcal{T}' \subseteq \mathcal{M}_2$. Since $\mathcal{M}_1, \mathcal{M}_2 \in mups(\mathcal{T}, C)$, we get $\mathcal{T}' = \mathcal{M}_1 = \mathcal{M}_2$ which contradicts with the assumption.

It is characteristic of our axiom pinpointing, in the sense to be made more precise, to uniquely identify each logical contradiction. For example, TBox $\mathcal{T} = \{A_1 \sqsubseteq A \sqcap B \sqcap A_2, A_2 \sqsubseteq \neg A \sqcap \neg B\}$, $\mathcal{T}$ is the only MUPS of A_1 while $mue(\mathcal{T}, A_1) = \{\{A_1 \sqsubseteq A, A_1 \sqsubseteq A_2, A_2 \sqsubseteq \neg A\}, \{A_1 \sqsubseteq B, A_1 \sqsubseteq A_2, A_2 \sqsubseteq \neg B\}\}$, which a MUPS has two MUE corresponding to. In this regard, entaiment-based axiom pinpointing is an extension of axiom pinpointing that MUE covers also the same unsatisfiable reasons of MUPS.

3.2 Syntactic Decomposition for Atomic Entailment

As previously mentioned, the theory of entailment-based axiom pinpointing is built on atomic entailment. For an incoherent terminology, we need to know the atomic entailments of each axiom instead. We give a syntactic decomposition notion to achieve this goal.

We give an overview of different kind of transformations that calculate the set of atomic entailment for an axiom in a terminology. Given a terminology $\mathcal{T}$ and an axiom $\alpha : C \sqsubseteq D$ where C is a atomic concept, apply the following transformation rules to α in each step (all rules of each step are correct[3]):

Step 1: (GCIs) Considering all such axioms $C_1 \sqsubseteq D_1, ..., C_n \sqsubseteq D_n$ in $\mathcal{T}$ where $C_i(1 \leq i \leq n)$ is a complex description, let $D' = (\neg C_1 \sqcup D_1) \sqcap ... \sqcap (\neg C_n \sqcup D_n)$, do $C' \sqsubseteq D \sqcap D'$, then transform D and D', respectively.
Step 2: (*Negation normal form*, NNF) Push all negation signs as far as possible into the description, using de Morgan's rules and usual rules for quantifiers[4].
Step 3: Repeated use of *distributive law*: $C_1 \sqcup (C_2 \sqcap C_3) = (C_1 \sqcup C_2) \sqcap (C_1 \sqcup C_3)$, $\forall R.(C_1 \sqcup (C_2 \sqcap C_3)) = \forall R.((C_1 \sqcup C_2) \sqcap (C_1 \sqcup C_3))$, $\exists R.(C_1 \sqcap (C_2 \sqcup C_3)) = \exists R.((C_1 \sqcap C_2) \sqcup (C_1 \sqcap C_3))$.
Step 4: Repeated use of following rules: $\forall R.(C_1 \sqcap C_2) = \forall R.C_1 \sqcap \forall R.C_2$, $\exists R.(C_1 \sqcup C_2) = \exists R.C_1 \sqcup \exists R.C_2$.

The transformation process always terminates and we end up with $D = D_1 \sqcap ... \sqcap D_m$ and $D' = D'_1 \sqcap ... \sqcap D'_n$ where constructor $\sqcap$ can only appear in $\lambda R.Y$ of $D_i(1 \leq i \leq m)$ and $D'_j(1 \leq j \leq n)$ while λ is constructor $\exists$, $\geq n$, or $\leq n$. Therefor, for any model $\mathcal{I}$, $\mathcal{I} \vDash \{\alpha : C \sqsubseteq D\} \Rightarrow \mathcal{I} \vDash C \sqsubseteq D_i(1 \leq i \leq m)$, and $C \sqsubseteq D_i$ has no entailment but itself. Consequently, $\{C \sqsubseteq D_1, ..., C \sqsubseteq D_m\}$ is the set of atomic entailment of α. Similarly, $\{C \sqsubseteq D_1, ..., C \sqsubseteq D_m, C \sqsubseteq D'_1, ..., C \sqsubseteq D'_n\}$ is the set of atomic entailment of α in terminology $\mathcal{T}$. Both the result of syntactic decomposition and the axiom have the same name and base symbols. Moreover, Since the result is obtained by a sequence of replacement steps, i.e., by replacing equals by equals. Therefore, $\mathcal{E}(\alpha)$ and α are syntactically and semantically equivalent, i.e., the result is the set of all atomic entailments of α. The atomic entailments of a terminology can be calculated by merging all axioms's. In example TBox $\mathcal{T}_1$, $\mathcal{E}(\alpha_1) = \{A_1 \sqsubseteq A_2,\ A_1 \sqsubseteq \exists r.A_2,\ A_1 \sqsubseteq A_3\}$.

On the other hand, using a rule $\mathcal{L} = \mathcal{R}$ above, it means $\mathcal{R}$ is obtained from $\mathcal{L}$. We can mark $\mathcal{R}$'s label is $\mathcal{L}$. Thus, keeping track of the transformations that occur during the processing step i.e. we can pinpoint the position of atomic entailment in original axiom.

[3] All these rules are correct and have been proved in the Description Logic Handbook[15].

[4] $\neg(\neg A) = A, \neg(\exists R.A) = \forall R.\neg A, \neg(\forall R.A) = \exists R.\neg A, \neg(\geq nR.A) = \leq (n-1)R.A, \neg(\leq nR.A) = \geq (n+1)R.A, \neg(C_1 \sqcup C_2) = \neg C_1 \sqcap \neg C_2, \neg(C_1 \sqcap C_2) = \neg C_1 \sqcup \neg C_2$.

4 Algorithms for Entailment-based Axiom Pinpointing

In this section, we discuss the algorithm for finding all MUE of an unsatisfiable concept. The algorithm we provide is reasoner-independent, in the sense that the DL reasoner is solely used as an oracle to determine concept satisfiability w.r.t a terminology. we provide the formal specification of the algorithm.

The ALL_MUE($\mathcal{T}, C, M, E$) algorithm receives a local terminology $\mathcal{T}$, a concept C, a local conflict M and a set of axioms E related to M directly[5], and outputs the set of all minimal subsets $\mathcal{T}' \subseteq \mathcal{E}(\mathcal{T} \cup E)$ such that C is unsatisfiable in $\mathcal{T}' \cup M$. The algorithm works in three main steps: first, it utilizes CONFLICT_HST to computes all related minimal contradiction of M from E for C; Then recursive call to the algorithm with the new parameters $\mathcal{T}$, M, and E for each related conflict we have obtained in previous step; Finally, combining the consequences of all recursive calls and obtain the final result. The loop in the second step is a main component of algorithm, which calculates the input parameters for next recursive call, it is mainly to do the following tasks: first of all, adding the obtained related conflict m to the original conflict M to get the new local conflict M'; Then, selecting axioms from $\mathcal{T}$ which is only related to the named symbols in m (because m has included all axioms related to M) dented by the new related axioms set A for the new local conflict M'; Last, get a new terminology $\mathcal{T}''$ by removing A from $\mathcal{T}$.

We can get all MUE of C in terminology $\mathcal{T}$ by calling ALL_MUE($\mathcal{T}, C, \varnothing, \varnothing$). Thus, the algorithm process guarantees three points as follows:

(a) Both axioms and named symbols of the input terminologies $\mathcal{T}$, M and E are mutually disjoint.
(b) $M \subseteq \mathcal{M}$ for every $\mathcal{M} \in mue(\mathcal{T} \cup M \cup E, C)$ if C is unsatisfiable in $\mathcal{T} \cup M \cup E$.
(c) C is satisfiable in $\mathcal{T} \cup M$ if M is not a MUE for C in $\mathcal{T} \cup M \cup E$.

Theorem 2. *Given an unsatisfiable concept C in a terminology $\mathcal{T}$, R returned by* ALL_MUE($\mathcal{T}, C, \varnothing, \varnothing$) *is the set of all* MUE *for C.*

Theorem 3. *The* CONFLICT_HST($\mathcal{T}, C, M, E$) *algorithm output all minimal subset $E' \subseteq E$ such that C is unsatisfiable in $\mathcal{T} \cup M \cup E'$, and C is satisfiable in $\mathcal{T} \cup M \cup E''$ for every $E'' \subset E'$.*

The CONFLICT_HST($N, F, HS, C, \mathcal{T}, M, E$) algorithm generates a Hitting Set Tree [16] with root node N, where a set F of conflict sets and a set HS of Hitting Sets are global, and outputs F. Initially, N, F and HS are empty, calling SINGLE_CONFLICT algorithm to get a value r for root node N if r is not empty. Then, generates the HST with root node N. In the loop, the algorithm generates a new node N' and a new edge e links N and N' in each iteration. Calling SINGLE_CONFLICT algorithm to obtain a value for the new node, and we mark the new node with $'\surd'$ if the value is empty.

[5] We say a TBox $\mathcal{T}'$ is directly related to the TBox $\mathcal{T}$ if all named symbols of $\mathcal{T}'$ is a subset of the signatures of $\mathcal{T}$.

```
**Algorithm:** ALL_MUE($\mathcal{T}, C, M, E$)
**Input:** a terminology $\mathcal{T}$, a concept $C$, a terminology $M$, a terminology $E$
**Output:** all minimal subsets of $\mathcal{E}(\mathcal{T} \cup E)$ conflict with $M$ w.r.t. $C$ $R$
$R \leftarrow \varnothing$;
$\mathcal{T}' \leftarrow \mathcal{T}$;
$E' \leftarrow E$;
if $M = \varnothing$ then                    /* The algorithm start with $M = \varnothing$ */
    $A \leftarrow \{\alpha \in \mathcal{T} \mid \alpha$ has the form $C \sqsubseteq D\}$;
    $\mathcal{T}' \leftarrow \mathcal{T} - A$;
    $E' \leftarrow \mathcal{E}(A)$;
if $C$ is unsatisfiable in $M$ then                    /* $M$ is a MUE of $C$ */
    $R \leftarrow \{M\}$;
    return $R$;
$H \leftarrow \varnothing$;
CONFLICT_HST($\varnothing, H, \varnothing, C, \mathcal{T}', M, E'$);
if $H = \varnothing$ then                    /* $H = \varnothing$ means $M$ is a MUE of $C$ */
    $R \leftarrow \{M\}$;
    return $R$;
for $m \in H$ do
    $M' \leftarrow M \cup m$;                    /* update the current MUE $M$ of $C$ */
    $S \leftarrow Sig(m) - \mathcal{D}(M') - \mathcal{B}(\mathcal{T}' \cup M')$;/*Select the related symbols of $M'$*/
    $A \leftarrow \{\alpha \in \mathcal{T}' \mid \alpha$ has the form $C' \sqsubseteq D'$ where $C' \in S\}$;
    $\mathcal{T}'' \leftarrow \mathcal{T}' - A$;
    $R' \leftarrow$ ALL_MUE($\mathcal{T}'', C, M', \mathcal{E}(A)$);
    $R \leftarrow R \cup R'$;
return $R$;
```

Two pruning strategy to the algorithm in order to reduce the size of HST and eliminate extraneous satisfiability tests. One is closing, if there exists a Hitting Set h in HS such that the path of N' is a superset of h, close node N' and the value is not computed for N' nor are any succor nodes generated, as indicted by a $'\times'$. The other one is reusing nodes: if there exists a node value k in F such that k and the path of N' are disjoint, set k as the value of N' directly without recalculation.

The SINGLE_CONFLICT($\mathcal{T}, C, M, E$) algorithm outputs a subset of E. In the loop, the algorithm removes an axiom from E in each iteration and check whether the concept C is satisfiable w.r.t. $\mathcal{T} \cup M \cup E$, in which case the axiom is added to R and reinserted into E. The process continues until all axioms in E have been tested. Finally, R is returned as output.

Theorem 4. *The* SINGLE_CONFLICT($\mathcal{T}, C, M, E$) *algorithm output a minimal subset $R \subseteq E$ such that C is unsatisfiable in $\mathcal{T} \cup M \cup R$, and C is satisfiable in $\mathcal{T} \cup M \cup R'$ for every $R' \subset R$.*

Proof. Let R be the output of algorithm SINGLE_CONFLICT($\mathcal{T}, C, M, E$). If C is satisfiable in $\mathcal{T} \cup M \cup E$, we get $R = \varnothing$. Otherwise, C is unsatisfiable in $\mathcal{T} \cup M \cup R$ upon termination. Suppose there exists a subset $R' \subset R$ such that C is unsatisfiable in $\mathcal{T} \cup M \cup R'$. Then, removing the axiom in $R - R'$ after the

```
**Algorithm:** CONFLICT_HST($N, F, HS, \mathcal{T}, C, M, E$)
**Input:** a node $N$, a set of conflict sets $F$, a set of hitting sets $HS$,
        a terminology $\mathcal{T}$, a concept $C$, a terminology $M$, a terminology $E$
**Output:** input $F$
if $N = \varnothing$ then
    $r \leftarrow$ SINGLE_CONFLICT($\mathcal{T}, C, M, E$);
    if $r = \varnothing$ then
        return;
    $\mathcal{L}(N) \leftarrow r$;
    $F \leftarrow \{r\}$;
for $\alpha \in \mathcal{L}(N)$ do
    create a new node $N'$ and set $\mathcal{L}(N') \leftarrow \varnothing$;
    create a new edge $e =< N, N' >$ with $\mathcal{L}(e) \leftarrow \alpha$;
    if there exists a set $h \in HS$ s.t. $h \subseteq \mathcal{P}(N) \cup \{\alpha\}$ then
        $\mathcal{L}(N') \leftarrow' \times '$;
        continue;
    else if there exists a set $k \in F$ s.t. $k \cap \mathcal{P}(N) = \varnothing$ then
            $\mathcal{L}(N') \leftarrow k$;
            CONFLICT_HST($N', F, HS, \mathcal{T}, C, M, E - \{\alpha\}$);
    else
        $m \leftarrow$ SINGLE_CONFLICT($\mathcal{T}, C, M, E - \{\alpha\}$);
        if $m = \varnothing$ then
            $\mathcal{L}(N') \leftarrow' \surd '$;
            $HS \leftarrow HS \cup \{\mathcal{P}(N) \cup \{\alpha\}\}$;
        else
            $\mathcal{L}(N') \leftarrow m$;
            $F \leftarrow F \cup \{m\}$;
            CONFLICT_HST($N', F, HS, \mathcal{T}, C, M, E - \{\alpha\}$);
```

removal of R', we get C is satisfiable in $\mathcal{T} \cup M \cup R'$, which contradicts with the assumption.

```
**Algorithm:** SINGLE_CONFLICT($\mathcal{T}, C, M, E$)
**Input:** a terminology $\mathcal{T}$, a concept $C$, a terminology $M$, a terminology$E$
**Output:** a set of axioms (a subset of $E$) $R$
$R \leftarrow \varnothing$;
$\mathcal{T}' \leftarrow \mathcal{T} \cup M \cup E$;
if ($C$ is satisfiable in $\mathcal{T}'$) then
    return $R$;
for $\alpha \in E$ do
    $\mathcal{T}' \leftarrow \mathcal{T}' - \{\alpha\}$;
    if $C$ is satisfiable in $\mathcal{T}'$ then
        $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{\alpha\}$;
        $R \leftarrow R \cup \{\alpha\}$;
return $R$;
```

The problem of finding minimal Hitting Sets is known to be NP-COMPLETE, our algorithm is associated with the size of the element in $mue(\mathcal{T}, C)$. In this case, Let n be the cardinality of $mue(\mathcal{T}, C)$ and $S = \{k_1, ..., k_n\}$ be the value set of the size of its elements, the number of calls to SINGLE_CONFLICT and satisfiability tests involved is at most $k_1 \cdot ... \cdot k_n$.

5 Evaluation

Our algorithms for fine-grained axiom pinpointing have been realized in JAVA (JDK 1.6)using Pellet as the black-box reasoner. Tests are performed on a standard Windows operating system (Intel(R) Core(TM)i5-3470 CPU @ 3. 20GHz, 8. 00GB).

Before providing an evaluation of our algorithm, we briefly want to discuss a case study from Pizza. Then, we give the experimental results of common ontologies.

Example 2. $IceCream$ is an unsatisfiable concept in ontology $Pizza$.

$IceCream \sqsubseteq Food \sqcap \exists hasTopping.FruitTopping$	disjoint$(IceCream, Pizza)$
$FruitTopping \sqsubseteq PizzaTopping$	disjoint$(IceCream, PizzaTopping)$
$PizzaTopping \sqsubseteq Food$	role $hasTopping$: domain $Pizza$

$IceCream$ in $Pizza$ ontology has only one MUE $\mathcal{M}$:
$\mathcal{M} = \{IceCream \sqsubseteq \exists hasTopping.FruitTopping$, disjoint$(IceCream, Pizza)$, role $hasTopping$: domain $Pizza\}$

For unsatisfiable concept $IceCream$, taking away any single axiom from $\mathcal{M}$ makes $IceCream$ satisfiable, while $\mathcal{M}$ is not a MUPS since $IceCream \sqsubseteq \exists hasTopping.FruitTopping$ is only a part of original axiom of $IceCream$, which pinpoint the accurate component of contradiction within the axioms. As a consequence, the MUE indeed helped in some cases.

We have performed some preliminary experiments. We evaluated the method on five real-life OWL-DL ontologies vary in size, complexity and expressivity: Koala, MadCow, Pizza, MGED, DICE. The basic information of our experimental ontologies are depicted in Table 1, the results of test ontologies are summarized in Table 2.

According to Table 1 and Table 2, the results show that the scale of ontology and number of unsatisfiable concepts introduce an increase in the running time w.r.t. the fine-grained axiom pinpointing procedure. In the case of Koala and MadCow ontology, where the number of axioms related to an unsatisfiable concept are small (less than 10), the program ends in a very short period of time. However, for DICE ontology, where axioms responsible for an unsatisfiable concept are large in number (nearly 100), the running time of procedure is longer.

Table 1. The characteristics of test ontologies.

$\mathcal{T}$	$\mathcal{L}(\mathcal{T})$	$\lvert\mathcal{D}(\mathcal{T})\rvert$	$\lvert\mathcal{B}(\mathcal{T})\rvert$	$\lvert\mathbf{R}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert U_T\rvert$
Koala	ALCHON(D)	17	3	4	18	3
MadCow	ALCH(D)	40	13	16	41	1
Pizza	SHOIN	99	0	6	103	2
MGED	ALCH	231	2	110	231	32
DICE	ALCH	505	22	5	505	76

Note. Columns are: the terminology $\mathcal{T}$, the expressivity of termminology ($\mathcal{L}(\mathcal{T})$), number of named symbols ($\lvert\mathcal{D}(\mathcal{T})\rvert$), number of base symbols ($\lvert\mathcal{B}(\mathcal{T})\rvert$), number of roles ($\lvert\mathbf{R}\rvert$), number of axioms ($\lvert\mathcal{T}\rvert$), number of unsatisfiable concepts ($\lvert U_T\rvert$).

Table 2. The results of test ontologies.

$\mathcal{T}$	Koala	MadCow	Pizza	MGED	DICE
$\lvert$MUE($\mathcal{T}$)$\rvert$	3	1	2	57	79
Time(ms)	16	16	9	594	287177

Note. Rows are: the test terminology ($\mathcal{T}$), the sum of MUE of all unsatisfiable concepts in terminology ($\lvert$MUE($\mathcal{T}$)$\rvert$), the execution time of ALL_MUE algorithm for all unsatisfiable concepts in terminology where the unit is millisecond (Time).

6 Conclusion

In order to pinpoint debugging more accurately, we use the entailments to replace the corresponding axioms, then identify a minimal unsatisfiable subset of entailments for the new terminology. A new formal definition of MUE have be provided in this paper. At the same time, we presented a black-box pinpointing algorithm to solve it. Experimental results on common ontologies show that our axiom pinpointing provides incoherent terminology with more accurate incoherent reasons without losing contradictions masked by MUPS, and the performance of our algorithm is influenced by the size of related axioms directly. For future work, we plan to adopt the dependency between concepts, investigate different kinds of selection function, that hopefully improve the efficiency of entailment-based axiom pinpointing.

Acknowledgments. This work was supported in part by NSFC under Grant Nos. 61272208, 61133011, 41172294, 61170092; Jilin Province Science and Technology Development Plan under Grant Nos.201201011.

References

1. Lambrix, P., Qi, G.L., Horridge. M., (eds): Proceedings of the First International Workshop on Debugging Ontologies and Ontology Mappings, (Galway, Ireland), LECP 79, Linköping University Electronic Press (2012)
2. Baader, F., Rafael Peñaloza, R.: Automata-Based Axiom Pinpointing. In: 4th international joint conference on Automated Reasoning, pp. 226-241. Springer-Verlag Berlin, Heidelberg (2008)
3. Meyer,T., Lee, K., Booth,R., Pan, J.Z.: Finding Maximally Satisfiable Terminologies for the Description Logic ALC. In: 21st National Conference on Artificial Intelligence, pp. 269-274. AAAI press (2006)
4. Lam, S.C., Pan, J.Z., Sleeman, D., Vasconcelos, W.: A Fine-Grained Approach to Resolving Unsatisfiable Ontologies. In: 2006 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 428-434. IEEE Computer Society Washington (2006)
5. Schlobach, S., Cornet, R.: Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In: 18th international joint conference on Artificial intelligence, pp. 355-360. Morgan Kaufmann Publishers Inc (2003)
6. Schlobach, S.: Diagnosing Terminologies. In: 20th national conference on Artificial intelligence, pp. 670-675. AAAI Press (2005)
7. Schlobach, S., Huang, Z.S., Cornet, R., Harmelen, F.V.: Debugging Incoherent Terminologies. J. Automated Reasoning. 39(3), 317-349 (2007)
8. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL Ontologies. In: 14th International Conference on World Wide Web, pp. 633-640. ACM Press, New York (2005)
9. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.: Debugging Unsatisfiable Concepts in OWL Ontologies. J. Web Semantics. 3(4), 268-293 (2005)
10. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding All Justifications of OWL DL Entailments. In: 6th international The semantic web and 2nd Asian conference on Asian semantic web conference, pp. 267-280. Springer-Verlag Berlin, Heidelberg (2007)
11. Wang, H., Horridge, M., Rector, A., Drummond, N., Seidenberg. J.: Debugging OWL-DL Ontologies: A Heuristic Approach. In: 4th International Semantic Web Conference, pp. 745-757. Springer-Verlag (2005)
12. Du, J.F., Qi, G.L., Ji, Q.: Goal-Directed Module Extraction for Explaining OWL DL Entailments. In: 8th International Semantic Web Conference, pp. 163-179. Springer-Verlag (2009)
13. Haase, P., Qi, G.L.: An Analysis of Approaches to Resolving Inconsistencies in DL-based Ontologies. In: International Workshop on Ontology Dynamics, pp. 97-109 (2007)
14. Horridge, M., Parsia, B., Sattler, U.: Justification Masking in Ontologies. In: 13th International Conference on the Principles of Knowledge Representation and Reasoning (2012)
15. Baader, F., Calvanese, D., McGuinness, D., et al.: The Description Logic Handbook: Theory, Implementation and Application. Second Edition. Cambridge University Press (2007)
16. Reiter, R.: A Theory of Diagnosis from First Principles. J. Artificial Intelligence, 32(1), 57-95 (1987)
17. Haase, P., Harmelen, F.V., Huang, Z.S., Stuckenschmidt, H., Sure, Y.: A Framework for Handling Inconsistency in Changing Ontologies. In: 4th International Semantic Web Conference, pp. 353-367. Springer-Verlag Berlin, Heidelberg (2005)