

# Graph-based approaches to debugging and revision of terminologies in DL-Lite



Xuefeng Fu, Guilin Qi\*, Yong Zhang, Zhangquan Zhou

School of Computer Science and Engineering, Southeast University, Nanjing, China

## ARTICLE INFO

### Article history:

Received 8 July 2015

Revised 28 January 2016

Accepted 29 January 2016

Available online 4 March 2016

### Keywords:

DL-Lite

Ontology debugging

Ontology revision

Scoring function

Hitting set tree

## ABSTRACT

In this paper, we deal with the problem of debugging and revision of incoherent terminologies. Ontology debugging aims to provide the explanation of the causes of incoherence and ontology revision aims to eliminate the incoherence. For this purpose, we propose the graph-based approaches to deal with the debugging and revision of terminologies for a family of lightweight ontology languages, DL-Lite. First of all, we transform DL-Lite ontologies to graphs. To deal with the problem of ontology debugging, we calculate the minimal incoherence-preserving subsets (MIPS) of an ontology by computing the minimal incoherence-preserving path-pairs (MIPP) based on the transformed graph. To deal with the problem of ontology revision, we propose the notion of revision state which separates the terminology of an ontology into two disjoint sets: the set of wanted axioms and the set of unwanted axioms. We further define a revision operator based on the revision state. Afterward, two revision algorithms are proposed to instantiate the revision operator: one is based on a scoring function, and the other one is based on a hitting set tree. We implement these algorithms and conduct experiments of ontology debugging and ontology revision on several adapted real ontologies. The experimental results of ontology debugging show that our approach of calculating MIPS based on graph is efficient and outperforms the state of the art. The experimental results of ontology revision show that the algorithm based on a scoring function is more efficient than the algorithm based on a hitting set tree.

© 2016 Elsevier B.V. All rights reserved.

## 1. Introduction

Ontologies play an important roles in semantic web and it is also important in recommender systems and social network [1]. However, the development and maintenance of ontologies are complex and error-prone. Thus, logical inconsistencies can easily occur in real ontologies [2]. Ontology debugging [3], which aims to pinpoint the causes of inconsistencies, and ontology revision [4], which aims to eliminate the inconsistencies, have become the key issues in ontology engineering.

Up to now, many approaches of ontology debugging are proposed to compute minimal unsatisfiability preserving subsets (MUPS) of an ontology [3,5,6] or to compute minimal incoherence preserving subsets (MIPS) of an ontology [5,7]. MUPS is useful for relating a set of axioms to an unsatisfiable concept and MIPS is the minimal sub-TBox of  $\mathcal{T}$  which is incoherent. It is argued that MIPS is more useful than MUPS when repairing an ontology, since

every MIPS is a MUPS and removing one axiom from each MIPS can resolve the incoherence of an ontology.

Similarly, there also exist many approaches for ontology revision. Most works, such as those reported in [8,9], on ontology revision are based on the AGM theory [10]. An important principle of these works is minimal change, i.e., a revision approach should delete information from the original ontology as little as possible. In [8], a kernel revision operator is proposed based on an incision function that is used to select axioms from each MIPS. In [9], the authors propose a revision operator based on a trust-based incision function which aims to find minimal set of axioms that need to be removed to render a terminology logically correct. Nevertheless, the delete operation may lead to the loss of valuable information in an ontology.

Recently, there is an increasing interest in inconsistency handling in DL-Lite, which is a family of tractable description logics. In this paper, we provide graph-based approaches to debug and to revise DL-Lite ontologies. At first, we encode an ontology into a graph, and derive subsumption relations between two concepts (or roles) by reachability of two nodes. Then, all MIPSs can be calculated by backtracking pairs of nodes in the graph. Thus this approach helps to avoid the computation of all MUPSs of the

\* Corresponding author. Tel.: +86 13376069256; fax: +86 2552090910.

E-mail addresses: [xfu@seu.edu.cn](mailto:xfu@seu.edu.cn) (X. Fu), [gqi@seu.edu.cn](mailto:gqi@seu.edu.cn), [xfcn@126.com](mailto:xfcn@126.com) (G. Qi), [zhangyong@seu.edu.cn](mailto:zhangyong@seu.edu.cn) (Y. Zhang), [quanzz@seu.edu.cn](mailto:quanzz@seu.edu.cn) (Z. Zhou).

ontology w.r.t. all unsatisfiable concepts. Based on MIPS, we propose the notion of revision state, which separates the terminology of an ontology into two disjoint sets: the set of wanted axioms and the set of unwanted axioms. We further define a refinement revision operator based on the revision state. This operator aims to retrieve useful information from the set of unwanted axioms. Afterward, we propose two revision algorithms to instantiate the revision operator: one is based on a scoring function, and the other one is based on a hitting set tree. We then evaluate the performance of these algorithms by conducting experiments on some (adapted) real ontologies. The experimental results of ontology debugging show that our approach of calculating MIPS based on graph is efficient and outperforms the state of the art. The experimental results of ontology revision show that the algorithm based on a scoring function is more efficient than the algorithm based on a hitting set tree.

The rest of the paper is organized as follows: We first introduce the theoretical basis of DL-Lite and a method of graph construction in Section 2. We then present our graph-based debugging and revision approaches in Section 3. In Section 4, we present our algorithms to instantiate the revision operator. In Section 5, we evaluate our algorithms using some adapted ontologies. We discuss related works in Section 6. Finally, we conclude this paper in Section 7.

## 2. Preliminaries

### 2.1. DL-Lite family

In this subsection, we introduce DL-Lite<sub>FR</sub>, which is an important language in DL-Lite that stands out for its tractable reasoning and efficient query answering [11,12]. We start with the introduction of DL-Lite<sub>core</sub>, which is the core language for the DL-Lite family [13]. The complex concepts and roles of DL-Lite<sub>core</sub> are defined as follows:

(1)  $B \rightarrow A|\exists R$ , (2)  $R \rightarrow P|P^-$ , (3)  $C \rightarrow B|\neg B$ , (4)  $E \rightarrow R|\neg R$ , where  $A$  denotes an atomic concept,  $P$  an atomic role,  $B$  a basic concept, and  $C$  a general concept. A basic concept can be either an atomic concept or a concept of the form  $\exists R$ , where  $R$  denotes a basic role which can be either an atomic role or the inverse of an atomic role.

In DL-Lite<sub>core</sub>, an ontology  $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$  consists of a TBox  $\mathcal{T}$  and an ABox  $\mathcal{A}$ , where  $\mathcal{T}$  is a finite set of concept inclusion assertions of the form:  $B \sqsubseteq C$ ; and  $\mathcal{A}$  is a finite set of membership assertions of the form:  $A(a)$ ,  $P(a, b)$ . In this paper, ABoxes will not be considered. DL-Lite<sub>FR</sub> extends DL-Lite<sub>core</sub> with inclusion assertions between roles which are the form  $R \sqsubseteq E$  and functionality on roles (or on their inverses) which are the form  $(\text{funct } R)$  or  $(\text{funct } R^-)$ . To keep the logic tractable, whenever a role inclusion  $R_1 \sqsubseteq R_2$  appears in  $\mathcal{T}$ , neither  $(\text{funct } R_2)$  nor  $(\text{funct } R_2^-)$  can appear in it.

The semantics of DL-Lite is defined by an interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \mathcal{I})$  which consists of a non-empty domain set  $\Delta^{\mathcal{I}}$  and an interpretation function  $\mathcal{I}$ , which maps individuals, concepts and roles to elements of the domain, subsets of the domain and binary relations on the domain, respectively. The interpretation function can be extended to arbitrary concept (or role) descriptions and inclusion (or membership) assertions in a standard way [11]. Given an interpretation  $\mathcal{I}$  and an assertion  $\phi$ ,  $\mathcal{I} \models \phi$  denotes that  $\mathcal{I}$  satisfies  $\phi$ . An interpretation is called a *model* of an ontology  $\mathcal{O}$ , iff it satisfies each assertion in  $\mathcal{O}$ . An ontology  $\mathcal{O}$  logically implies an assertion  $\phi$ , written  $\mathcal{O} \models \phi$ , if all models of  $\mathcal{O}$  satisfy  $\phi$ .

In DL-Lite, the assertions of the form  $B_1 \sqsubseteq B_2$  or  $R_1 \sqsubseteq R_2$  are called positive inclusions (PIs) and the assertions of the form  $B_1 \sqsubseteq \neg B_2$  or  $R_1 \sqsubseteq \neg R_2$  are called negative inclusions (NIs). In order to take into account the interaction of inclusions, we introduce the definition of NI-closure [11].

**Definition 1.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox. We define the NI-closure of  $\mathcal{T}$ , denoted by  $\text{cln}(\mathcal{T})$ , inductively as follows:

1. All negative inclusion assertions (NIs) in  $\mathcal{T}$  are also in  $\text{cln}(\mathcal{T})$ .
2. All functionality assertions in  $\mathcal{T}$  are also in  $\text{cln}(\mathcal{T})$ .
3. If  $B_1 \sqsubseteq B_2$  is in  $\mathcal{T}$  and  $B_2 \sqsubseteq \neg B_3$  or  $B_3 \sqsubseteq \neg B_2$  is in  $\text{cln}(\mathcal{T})$ , then also  $B_1 \sqsubseteq \neg B_3$  is in  $\text{cln}(\mathcal{T})$ .
4. If  $R_1 \sqsubseteq R_2$  is in  $\mathcal{T}$  and  $\exists R_2 \sqsubseteq \neg B$  or  $B \sqsubseteq \neg \exists R_2$  is in  $\text{cln}(\mathcal{T})$ , then also  $\exists R_1 \sqsubseteq \neg B$  is in  $\text{cln}(\mathcal{T})$ .
5. If  $R_1 \sqsubseteq R_2$  is in  $\mathcal{T}$  and  $\exists R_2^- \sqsubseteq \neg B$  or  $B \sqsubseteq \neg \exists R_2^-$  is in  $\text{cln}(\mathcal{T})$ , then also  $\exists R_1^- \sqsubseteq \neg B$  is in  $\text{cln}(\mathcal{T})$ .
6. If  $R_1 \sqsubseteq R_2$  is in  $\mathcal{T}$  and  $R_2 \sqsubseteq \neg R_3$  or  $R_3 \sqsubseteq \neg R_2$  is in  $\text{cln}(\mathcal{T})$ , then also  $R_1 \sqsubseteq \neg R_3$  is in  $\text{cln}(\mathcal{T})$ .
7. If one of the assertions  $\exists R \sqsubseteq \neg \exists R$ ,  $\exists R^- \sqsubseteq \neg \exists R^-$  and  $R \sqsubseteq \neg R$  is in  $\text{cln}(\mathcal{T})$ , then all three such assertions are in  $\text{cln}(\mathcal{T})$ .

**Definition 2.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox. We define the PI-closure of  $\mathcal{T}$ , denoted by  $\text{clp}(\mathcal{T})$ , inductively as follows:

1. All positive inclusion assertions (PIs) in  $\mathcal{T}$  are also in  $\text{clp}(\mathcal{T})$ .
2. If  $B_1 \sqsubseteq B_2$  is in  $\text{clp}(\mathcal{T})$  and  $B_2 \sqsubseteq B_3$  is in  $\text{clp}(\mathcal{T})$ , then also  $B_1 \sqsubseteq B_3$  is in  $\text{clp}(\mathcal{T})$ .
3. If  $R_1 \sqsubseteq R_2$  is in  $\text{clp}(\mathcal{T})$  and  $R_2 \sqsubseteq R_3$  is in  $\text{clp}(\mathcal{T})$ , then also  $R_1 \sqsubseteq R_3$  is in  $\text{clp}(\mathcal{T})$ .
4. If  $R_1 \sqsubseteq R_2^-$  is in  $\text{clp}(\mathcal{T})$  and  $R_2 \sqsubseteq R_3^-$  is in  $\text{clp}(\mathcal{T})$ , then also  $R_1 \sqsubseteq R_3$  is in  $\text{clp}(\mathcal{T})$ .
5. If  $R_1 \sqsubseteq R_2$  is in  $\text{clp}(\mathcal{T})$  and  $\exists R_2 \sqsubseteq B$  is in  $\text{clp}(\mathcal{T})$ , then also  $\exists R_1 \sqsubseteq B$  is in  $\text{clp}(\mathcal{T})$ .
6. If  $R_1 \sqsubseteq R_2$  is in  $\text{clp}(\mathcal{T})$  and  $\exists R_2^- \sqsubseteq B$  is in  $\text{clp}(\mathcal{T})$ , then also  $\exists R_1^- \sqsubseteq B$  is in  $\text{clp}(\mathcal{T})$ .

Finally, let  $\text{cl}(\mathcal{T}) = \text{clp}(\mathcal{T}) \cup \text{cln}(\mathcal{T})$ , where  $\text{cl}(\mathcal{T})$  is the closure of  $\mathcal{T}$ . We call a TBox  $\mathcal{T}$  is closed if  $\mathcal{T} = \text{cl}(\mathcal{T})$ .

### 2.2. Logical conflict in TBox

We now introduce several important notations closely related to ontology revision given in [14]. These notations will be used in the sequel.

**Definition 3.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox. A concept  $C$  (resp. role  $R$ ) in  $\mathcal{T}$  is unsatisfiable if and only if for each model  $\mathcal{I}$  of  $\mathcal{T}$ ,  $C^{\mathcal{I}} = \emptyset$  (resp.  $R^{\mathcal{I}} = \emptyset$ ).

A TBox  $\mathcal{T}$  is incoherent if and only if there exist at least one unsatisfiable concepts or roles in  $\mathcal{T}$ .

**Definition 4.** 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 and only if  $\mathcal{T}'$  is incoherent and every sub-TBox  $\mathcal{T}'' \subset \mathcal{T}'$  is coherent.

**Example 1.** Given a TBox  $\mathcal{T}$ , where  $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C, B \sqsubseteq D, D \sqsubseteq \neg C, A \sqsubseteq \neg B\}$ . According to Definition 3,  $A$  and  $B$  are unsatisfiable concepts. According to Definition 4, there are two MIPSs in  $\mathcal{T}$ , one is  $\{A \sqsubseteq B, A \sqsubseteq \neg B\}$ , and the other is  $\{B \sqsubseteq C, B \sqsubseteq D, D \sqsubseteq \neg C\}$ .

**Definition 5.** Let  $\mathcal{T}$  be a coherent DL-Lite<sub>FR</sub> TBox. A TBox is logical closure of  $\mathcal{T}$ , denoted by  $\text{cn}(\mathcal{T})$ , if  $\phi \in \text{cn}(\mathcal{T})$  for any  $\mathcal{T} \models \phi$ .

In the following, we give the relationship between  $\text{cl}(\mathcal{T})$  and  $\text{cn}(\mathcal{T})$ .

**Theorem 1.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox,  $\text{cl}(\mathcal{T}) = \text{cn}(\mathcal{T})$  if  $\mathcal{T}$  is coherent.

**Proof.** According to the definition of  $\text{cn}(\mathcal{T})$ , it includes all axioms entailed by  $\mathcal{T}$ . Since for inductive rules in Definition 2 and Definition 1, all positive inclusion assertions, negative inclusion assertions and functionality assertions that can be entailed by  $\mathcal{T}$ , are included in  $\text{cl}(\mathcal{T})$ . Therefore,  $\text{cl}(\mathcal{T}) \subseteq \text{cn}(\mathcal{T})$ .

Each axiom  $\phi$  in a TBox of a DL-Lite ontology can have three forms: positive inclusion, negative inclusion and functionality assertions. To arbitrary axiom  $\phi \in cn(\mathcal{T})$ , we discuss the relationship between  $cn(\mathcal{T})$  and  $cl(\mathcal{T})$  by considering these three forms respectively.

Let  $\phi$  be a positive inclusion assertion of the form  $A \sqsubseteq B$ . Assuming that  $\mathcal{I}$  is a model of  $\mathcal{T}$ ,  $\mathcal{I}$  satisfies axiom  $A \sqsubseteq B$ , that is  $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ . Since  $\mathcal{T}$  is coherent and  $\phi \in cn(\mathcal{T})$  implies that  $\mathcal{T} \models A \sqsubseteq B$ , there exist concepts, without loss of generality,  $B_1, \dots, B_n$ , such that  $A^{\mathcal{I}} \subseteq B_1^{\mathcal{I}} \subseteq \dots \subseteq B_n^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ . Then we can get that  $\mathcal{T} \models A \sqsubseteq B_1$ ,  $\mathcal{T} \models B_1 \sqsubseteq B_2, \dots, \mathcal{T} \models B_{n-1} \sqsubseteq B_n$ ,  $\mathcal{T} \models B_n \sqsubseteq B$ . According to the rules of definition of  $clp(\mathcal{T})$  which entail  $A \sqsubseteq B$ . Therefore,  $\phi \in cl(\mathcal{T})$ .

Let  $\phi$  be a negative inclusion assertion or a functionality assertion.  $\phi \in cn(\mathcal{T})$  implies  $\mathcal{T} \models \phi$ . According to the Corollary 13 of [11], if  $\mathcal{T} \models \phi$  then  $\phi \in cln(\mathcal{T})$ . Hence we can infer that  $\phi \in cl(\mathcal{T})$  because  $cl(\mathcal{T}) = clp(\mathcal{T}) \cup cln(\mathcal{T})$ .

Finally, we conclude that  $cl(\mathcal{T}) = cn(\mathcal{T})$ .  $\square$

The coherence of TBox  $\mathcal{T}$  is the prerequisite for  $cl(\mathcal{T}) = cn(\mathcal{T})$ . If TBox  $\mathcal{T}$  is incoherent,  $\mathcal{T}$  include at least one unsatisfiable concepts or roles. An unsatisfiable concept or role can infer an arbitrary consequence, so that the reasoning would turn out to be trivial. It can be shown in Example 2.

**Example 2.** Given a TBox  $\mathcal{T}$ , where  $\mathcal{T} = \{A \sqsubseteq B, A \sqsubseteq \neg B, C \sqsubseteq D\}$ . It is easy to check that  $A$  is an unsatisfiable concept and  $\mathcal{T}$  is incoherent. According to Definition 5, we can get  $cn(\mathcal{T}) = \{A \sqsubseteq B, A \sqsubseteq \neg B, C \sqsubseteq D, A \sqsubseteq \neg A, A \sqsubseteq C, A \sqsubseteq D\}$ .

According to the definition of  $cl(\mathcal{T})$ , we can get  $cl(\mathcal{T}) = \{A \sqsubseteq B, A \sqsubseteq \neg B, C \sqsubseteq D, A \sqsubseteq \neg A\}$ .

### 2.3. Graph construction

Inspired by the work given in [15,16] and [17], we transform a DL-Lite ontology into a graph. Based on the constructed graph, we can propose our debugging and revision approaches, which will be discussed in the next section.

Let  $\mathcal{T}$  be a DL-Lite TBox over a signature  $\Sigma_{\mathcal{T}}$ , containing symbols for atomic elements, i.e., atomic concept and atomic roles. The digraph  $G_{\mathcal{T}} = \langle N, E \rangle$  constructed from  $\mathcal{T}$  over the signature  $\Sigma_{\mathcal{T}}$  is given as follows:

1. For each concept  $B$  in  $\Sigma_{\mathcal{T}}$ ,  $N$  contains the node  $B$ .
2. For each role  $P$  in  $\Sigma_{\mathcal{T}}$ ,  $N$  contains the nodes  $P, P^-, \exists P, \exists P^-$ .
3. For each concept inclusion  $B_1 \sqsubseteq B_2 \in \mathcal{T}$ ,  $E$  contains the arc( $B_1, B_2$ ).
4. For each concept inclusion  $B_1 \sqsubseteq \neg B_2 \in \mathcal{T}$ ,  $E$  contains the arc( $B_1, \neg B_2$ ) and  $N$  contains the node  $\neg B_2$ .
5. For each role inclusion  $R_1 \sqsubseteq R_2 \in \mathcal{T}$ ,  $E$  contains the arc( $R_1, R_2$ ), arc( $R_1^-, R_2^-$ ), arc( $\exists R_1, \exists R_2$ ), arc( $\exists R_1^-, \exists R_2^-$ ).
6. For each role inclusion  $R_1 \sqsubseteq \neg R_2 \in \mathcal{T}$ ,  $E$  contains the arc( $R_1, \neg R_2$ ), arc( $R_1^-, \neg R_2^-$ ), arc( $\exists R_1, \neg \exists R_2$ ), arc( $\exists R_1^-, \neg \exists R_2^-$ ) and  $N$  contains nodes  $\neg R_2, \neg R_2^-, \neg \exists R_2, \neg \exists R_2^-$ .

Based on (1)–(6), we construct a graph from a TBox  $\mathcal{T}$ . It has been shown in [16] that the problem of TBox classification in a DL-Lite ontology  $\mathcal{O}$  can be done by computing the transitive closure of graph  $G_{\mathcal{T}}$ . For simplicity, we call these rules **Construction Rules**, and we use  $C(R)$  to denote the node w.r.t. concept  $C$  (role  $R$ ).

In a graph built by Construction Rules, each node represents a basic concept or a basic role, while each arc represents an inclusion assertion, i.e. the start and end nodes are corresponding respectively to the left-hand and right-hand parts of an inclusion assertion. In order to ensure that the information represented in the TBox is preserved by the graph, we add nodes  $R, R^-, \exists R, \exists R^-$  for each role  $R$ , arc( $R_1, R_2$ ), arc( $R_1^-, R_2^-$ ), arc( $\exists R_1, \exists R_2$ ), arc( $\exists R_1^-, \exists R_2^-$ ) for each role inclusion assertion  $R_1 \sqsubseteq R_2$ .

### 3. Graph-based debugging and revision

In this section, we introduce our graph-based approaches for TBox debugging and revision. At first, we give theorems that serve as the theoretical basis for our approach.

#### 3.1. Theoretical basis

To explain the equivalence of a DL-Lite TBox and its corresponding graph, we firstly introduce the theorem given in [16].

**Theorem 2.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox that contains only positive inclusions and let  $S_1$  and  $S_2$  be two atomic concepts, or two atomic roles.  $S_1 \sqsubseteq S_2$  is entailed by  $\mathcal{T}$  if and only if at least one of the following conditions holds:

1. a set  $\mathcal{P}$  of positive inclusions exists in  $\mathcal{T}$ , such that  $\mathcal{P} \models S_1 \sqsubseteq S_2$
2.  $\mathcal{T} \models S_1 \sqsubseteq \neg S_1$ .

We can extend Theorem 2 to general concepts or roles, i.e., the subsumption relation may be a negative inclusion.

**Theorem 3.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox and let  $S_1$  and  $S_2$  be two atomic concepts, or two atomic roles.  $S_1 \sqsubseteq \neg S_2$  is entailed by  $\mathcal{T}$  if and only if at least one of the following conditions holds:

1. a set  $\mathcal{P}$  of positive inclusions and a negative inclusion  $n \in \mathcal{T}$  exist in  $\mathcal{T}$ , such that  $\mathcal{P} \cup \{n\} \models S_1 \sqsubseteq \neg S_2$
2.  $\mathcal{T} \models S_1 \sqsubseteq \neg S_1$ .

**Proof.** ( $\Leftarrow$ ) This can be easily proved.

( $\Rightarrow$ ) Assume  $\mathcal{T} \models S_1 \sqsubseteq \neg S_2$ . Without loss of generality, we suppose that there exists a minimal axiom set  $\mathcal{T}' = \{\phi_1, \phi_2, \dots, \phi_n\}$  in  $\mathcal{T}$  and  $\mathcal{T}' \models S_1 \sqsubseteq \neg S_2$ . According to Definition 1 and Definition 2, negative inclusions do not concur in the entailment of a set of positive inclusions. Therefore, there exists at least one negative inclusions in  $\mathcal{T}'$ . Let  $\phi_i$  be the first negative inclusion that occurs in the  $\mathcal{T}'$  and is of the form  $B_1 \sqsubseteq \neg B_2$ , then the following cases are considered:

1.  $\phi_i$  is the first negative inclusion of  $\mathcal{T}'$ , then  $\phi_1, \phi_2, \dots, \phi_{i-1}$  are positive inclusions. If there has no negative inclusions in the  $\phi_{i+1}, \phi_{i+2}, \dots, \phi_n$ , we can conclude that  $\mathcal{T}'$  only contains one negative inclusion  $\phi_i$ .
2. Suppose that there have more than one negative inclusions in  $\phi_{i+1}, \phi_{i+2}, \dots, \phi_n$ , we select one negative inclusion and assume that it is of the form  $B'_1 \sqsubseteq \neg B'_2$ . According to the syntax of DL-Lite [11], the general concept (role) only occurs on the right-hand side of inclusion assertions. Due to this constraint, inference cannot be carried out between  $\phi_i$  (we assume the form of  $\phi_i$  is  $B_1 \sqsubseteq \neg B_2$ ) and  $B'_1 \sqsubseteq \neg B'_2$ , so  $\mathcal{T}'$  is not a minimal set that entails  $S_1 \sqsubseteq \neg S_2$ . Then we get a contradiction.

$\square$

Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox and let  $G_{\mathcal{T}} = \langle N, E \rangle$  be the digraph constructed from  $\mathcal{T}$  according to the Construction Rules. We denote the transitive closure of  $G_{\mathcal{T}}$  by  $G_{\mathcal{T}}^* = \langle N, E^* \rangle$ . According to Theorem 2 and Theorem 3, we can see that the entailment problem can be reduced to the graph reachability problem between concepts and roles. Consider for example,  $\mathcal{T} = \{B_1 \sqsubseteq B_2, B_2 \sqsubseteq B_3\}$ , whose corresponding graph  $G$  contains two edges arc( $B_1, B_2$ ) and arc( $B_2, B_3$ ), it is obvious that  $B_1 \sqsubseteq B_3 \in cl(\mathcal{T})$  and arc( $B_1, B_3$ ) belongs to the transitive closure of  $G$ .

In order to prove the equivalence of an ontology and the graph constructed from it w.r.t. classification, we cite a theorem given in [16].

**Theorem 4.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox that contains only positive inclusions and let  $G_{\mathcal{T}} = \langle N, E \rangle$  be its digraph representation. Let  $S_1, S_2$

be two basic concepts (roles).  $S_1 \sqsubseteq S_2 \in cl(\mathcal{T})$  if and only if  $arc(S_1, S_2) \in E^*$ .

Then we extend [Theorem 4](#) as follows.

**Theorem 5.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox and let  $G_{\mathcal{T}} = \langle N, E \rangle$  be the digraph constructed from  $\mathcal{T}$  according to the Construction Rules. Let  $m$  be a basic concept (role) and  $n$  be a general concept (role).  $m \sqsubseteq n \in cl(\mathcal{T})$  if and only if  $arc(m, n) \in E^*$ .

**Proof.** ( $\Leftarrow$ ) If  $arc(m, n) \in E^*$ , according to the definition of transitive closure of graph, there exists at least one paths starting from the node  $m$  to the node  $n$  in  $G_{\mathcal{T}}$ . Moreover, according to Construction Rules, for each edge in  $G_{\mathcal{T}}$ , it corresponds to at least one inclusion assertions in  $\mathcal{T}$  or  $cl(\mathcal{T})$ . We assume  $\mathcal{S}$  is the set of these assertions. Obviously that  $\mathcal{S} \models m \sqsubseteq n$ . Therefore,  $m \sqsubseteq n \in cl(\mathcal{T})$ .

( $\Rightarrow$ ) If  $m \sqsubseteq n \in cl(\mathcal{T})$ , we consider the following two cases:

1. If  $m \sqsubseteq n \in clp(\mathcal{T})$ , we can infer  $arc(m, n) \in E^*$  based on [Theorem 4](#).
2. If  $m \sqsubseteq n \in cln(\mathcal{T})$ , then  $m \sqsubseteq n$  is in the form of  $B_1 \sqsubseteq \neg B_2$ . According to [Theorem 3](#), there exist a set of PIs and a NI  $n$  such that  $P \cup \{n\} \models B_1 \sqsubseteq \neg B_2$ . Without loss of generality, we assume  $n = \{B' \sqsubseteq \neg B_2\}$  and  $P \models B_1 \sqsubseteq B'$ . According to [Theorem 2](#), we can infer that  $arc(B_1, B') \in E^*$ . Therefore,  $arc(B_1, \neg B_2) \in E^*$ .

□

### 3.2. TBox debugging

In order to revise an incoherent TBox, we propose an approach for TBox debugging which provides the explanation of the incoherence of an TBox. In this subsection we present our graph-based approach for TBox debugging. At first, we define the notion of MIPP which corresponds to MIPS of an incoherent TBox.

**Definition 6.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox and  $G_{\mathcal{T}} = \langle N, E \rangle$  be the digraph constructed from  $\mathcal{T}$  according to the Construction Rules. Then for arbitrary node  $C \in N$ , if there exist two paths that start from node  $C$  to node  $D$  and from node  $C$  to node  $\neg D$  respectively in  $G_{\mathcal{T}}$  and there does not exist joint edge between path  $C \rightarrow D$  and path  $C \rightarrow \neg D$ , we call these two paths as minimal incoherence-preserving path-pair (MIPP).

We use  $mipp(G_{\mathcal{T}})$  to denote the set of all MIPPs in  $G_{\mathcal{T}}$ .

In the following, we show that the problem of computing MIPS of  $\mathcal{T}$ , denoted by  $mips(\mathcal{T})$ , can be reduced to that of finding MIPP in graph  $G_{\mathcal{T}}$ . Based on the correspondence between the syntax of DL-lite and the graph representation, the following theorem holds.

**Theorem 6.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox,  $\mathcal{T}$  is incoherent if and only if there exist at least one MIPP in  $G_{\mathcal{T}}$ .

**Proof.** ( $\Leftarrow$ ) If there exists a MIPP in  $G_{\mathcal{T}}$ , according to [Definition 6](#), we assume that there exist two paths, one is  $C \rightarrow D$ , and the other is  $C \rightarrow \neg D$ . In this case, we can infer both  $C \sqsubseteq D$  and  $C \sqsubseteq \neg D$  from  $\mathcal{T}$  by [Theorem 5](#). Therefore,  $\mathcal{T}$  contains at least one unsatisfiable concepts or roles.

( $\Rightarrow$ )  $\mathcal{T}$  is incoherent so that there will be at least one unsatisfiable concepts or roles in  $\mathcal{T}$ . Without loss of generality, suppose  $C$  is an unsatisfiable concept and  $C \sqsubseteq D \in cl(\mathcal{T})$ ,  $C \sqsubseteq \neg D \in cl(\mathcal{T})$ . According to [Theorem 5](#), we have  $arc(C, D) \in G_{\mathcal{T}}$  and  $arc(C, \neg D) \in G_{\mathcal{T}}$ . It is easy to see that there exist two paths without joint edge in the digraph  $G_{\mathcal{T}}$ , that is a MIPP by [Definition 6](#). □

We use  $\mathcal{M}$  to denote the set of inclusion assertions in a path  $P$  on the graph. It is obvious  $\mathcal{M} \subseteq cl(\mathcal{T})$ . According to [Theorem 3](#) and [Theorem 5](#), there will be a minimal subset  $\mathcal{S} \subseteq \mathcal{T}$  such that  $\mathcal{S} \models \mathcal{M}$  and for every  $\mathcal{S}' \subset \mathcal{S}$ ,  $\mathcal{S}' \not\models \mathcal{M}$ . We call  $\mathcal{S}$  as the minimal set in  $\mathcal{T}$

corresponding to a path  $P$ , denoted by  $\mathcal{S}_{\mathcal{T} \rightarrow P}$ . Based on the notion, we discuss the relationship between MIPP and MIPS.

**Theorem 7.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox and  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP}$  be the minimal set in  $\mathcal{T}$  corresponding to a MIPP in  $G_{\mathcal{T}}$ . Then  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP}$  is a MIPS in  $\mathcal{T}$ .

**Proof.** By [Theorem 6](#), it is easy to see that  $\mathcal{T}$  is incoherent. Let  $\mathcal{M}$  be the set of inclusion assertions corresponding to the MIPP. For any  $\mathcal{S}' \subset \mathcal{S}_{\mathcal{T} \rightarrow MIPP}$ ,  $\mathcal{S}' \not\models \mathcal{M}$  because  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP}$  is the minimal subset of  $\mathcal{T}$  that satisfies  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP} \models \mathcal{M}$ . According to Construction Rules, there does not exist a MIPP in  $G_{\mathcal{S}'}$ . Then  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP}$  is incoherent and  $\mathcal{S}'$  is coherent. According to [Definition 4](#),  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP}$  is a MIPS in  $\mathcal{T}$ . □

**Theorem 8.** Let  $\mathcal{T}$  be a DL-Lite<sub>FR</sub> TBox. The set of all  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP}$  in  $G_{\mathcal{T}}$  corresponds to the set of all MIPS in  $\mathcal{T}$ .

**Proof.** For any MIPP in  $mipp(G_{\mathcal{T}})$ , by [Theorem 7](#),  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP} \in mips(\mathcal{T})$ . For any MIPS  $\mathcal{M}$  in  $mips(\mathcal{T})$ ,  $\mathcal{M}$  is incoherent and for any  $\mathcal{M}' \subset \mathcal{M}$ ,  $\mathcal{M}'$  is coherent. Therefore, there exists a MIPP in  $G_{\mathcal{M}}$  and it does not exist in  $G_{\mathcal{M}'}$ . It is easy to see that  $\mathcal{M}$  corresponds to the MIPP in  $G_{\mathcal{T}}$  and  $\mathcal{M}$  is contained in the set of all  $\mathcal{S}_{\mathcal{T} \rightarrow MIPP}$  corresponding to MIPP in  $\mathcal{T}$ .

The case of role  $R$  can be proved analogously. □

### 3.3. TBox revision

TBox debugging only provides the cause of incoherence in a TBox. We now consider TBox revision which helps to further eliminate the conflicts based on the result of debugging.

In our work, we restrict to the case that  $\mathcal{T}_1$  and  $\mathcal{T}_2$  are coherent and  $\mathcal{T}_1 \cup \mathcal{T}_2$  is incoherent. We consider extending the definition of MIPS given in [\[8\]](#).

**Definition 7.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two coherent DL-Lite<sub>FR</sub> TBoxes and  $\mathcal{T}_1 \cup \mathcal{T}_2$  is incoherent. A minimal incoherence-preserving sub-TBox of  $\mathcal{T}_1$  w.r.t.  $\mathcal{T}_2$ , denoted by  $MIPS_{\mathcal{T}_2}(\mathcal{T}_1)$ , is a sub-TBox of  $\mathcal{T}_1$  which satisfies both of the following conditions:

1.  $MIPS_{\mathcal{T}_2}(\mathcal{T}_1) \cup \mathcal{T}_2$  is incoherent.
2.  $\forall \mathcal{T}'_1 \subset MIPS_{\mathcal{T}_2}(\mathcal{T}_1)$ ,  $\mathcal{T}'_1 \cup \mathcal{T}_2$  is coherent.

We denote the set of all MIPS of  $\mathcal{T}_1$  w.r.t  $\mathcal{T}_2$  by  $mips_{\mathcal{T}_2}(\mathcal{T}_1)$

[Definition 7](#) can be applied to a single TBox. In this case, an incoherent TBox can be divided into a static part and a rebuttal part. The rebuttal part consists of the axioms in MIPS. In the following, we present a modified notation of revision state which was proposed in [\[18\]](#).

**Definition 8.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two DL-Lite<sub>FR</sub> TBoxes and  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . The revision state of  $\mathcal{T}$  is a tuple  $(\mathcal{T}, \mathcal{T}_{\models}, \mathcal{T}_{\not\models})$  that satisfies the following conditions:

1.  $\mathcal{T}_{\models} \subseteq \mathcal{T}$ ,  $\mathcal{T}_{\not\models} \subseteq \mathcal{T}_1$ .
2.  $\mathcal{T}_{\models} \cap \mathcal{T}_{\not\models} = \emptyset$ .

A revision state is complete, if  $\mathcal{T} = \mathcal{T}_{\models} \cup \mathcal{T}_{\not\models}$ , and incomplete otherwise. A revision state is coherent if there is no  $\phi \in \mathcal{T}_{\not\models}$  such that  $\mathcal{T}_{\models} \models \phi$ .

A revision state actually separates a TBox into two parts:  $\mathcal{T}_{\models}$ , which includes wanted inclusion assertions, and  $\mathcal{T}_{\not\models}$ , which contains unwanted inclusion assertions. In a complete revision state, we can remove axioms in unwanted set  $\mathcal{T}_{\not\models}$  from  $\mathcal{T}$  to accomplish the purpose of revision. The following example illustrates the idea.

**Example 3.** Let  $\mathcal{T}_1 = \{A \sqsubseteq \exists S, \exists S \sqsubseteq E\}$ ,  $\mathcal{T}_2 = \{\exists R \sqsubseteq X, A \sqsubseteq \neg X, S \sqsubseteq R\}$ .  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . It is easy to check that concept  $A$  is an unsatisfiable concept in  $\mathcal{T}$ , and we get a complete revision state of  $\mathcal{T}$  given as follows.



1.  $\mathcal{T}_\mathbb{E} = \{\exists S \sqsubseteq E, \exists R \sqsubseteq X, A \sqsubseteq \neg X, S \sqsubseteq R\}$ .
2.  $\mathcal{T}_\mathbb{E} = \{A \sqsubseteq \exists S\}$ .

We can simply revise the merged TBox by removing  $\mathcal{T}_\mathbb{E}$ . However, the naive revision method may lose too much information in the original TBox. In Example 3, we can see that  $A \sqsubseteq E \in cl(\mathcal{T})$  and  $A \sqsubseteq E \notin cl(\mathcal{T}_\mathbb{E})$ . Nevertheless,  $\{A \sqsubseteq E\} \cup \mathcal{T}_\mathbb{E}$  is coherent, which means the inclusion assertion  $\{A \sqsubseteq E\}$  can be preserved. In order to avoid the loss of information, we define the refinement of revision state.

**Definition 9.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two DL-Lite<sub>FR</sub> TBoxes,  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . A coherent refinement of revision state  $(\mathcal{T}, \mathcal{T}_\mathbb{E}, \mathcal{T}_\mathbb{E}')$  is  $(\mathcal{T}, \mathcal{T}_\mathbb{E} \cup \mathcal{T}_\mathbb{E}', \mathcal{T}_\mathbb{E}')$ , where  $\mathcal{T}_\mathbb{E}' = \{\psi \in cl_\mathcal{T}(\phi) | \mathcal{T}_\mathbb{E} \cup \{\psi\} \text{ is coherent, } \phi \in \mathcal{T}_\mathbb{E}\}$ .

In the above definition,  $cl_\mathcal{T}(\phi)$  denotes the closure of inclusion assertion  $\phi$  w.r.t  $\mathcal{T}$ . Referring to the previous rules in Definition 2 and Definition 1, we give inductive rules for calculating  $cl_\mathcal{T}(\phi)$ . At first, inclusion assertions  $\phi$  is in  $cl_\mathcal{T}(\phi)$ . If  $B_1 \sqsubseteq B_2$  is in  $cl_\mathcal{T}(\phi)$  and  $B_2 \sqsubseteq B_3$  is in  $\mathcal{T}$ , then  $B_1 \sqsubseteq B_3$  is in  $cl_\mathcal{T}(\phi)$ . Analogously, other rules can be defined.

When incorporating a new TBox into an old one, refinement of revision state can retain original information as much as possible by adding some axioms into  $\mathcal{T}_\mathbb{E}$ . According to Definition 9, in Example 3, we can obtain  $\mathcal{T}_\mathbb{E}' = \{A \sqsubseteq E\}$ .

In general, there could be more than one revision states. A problem is that which axioms are unwanted. Therefore, we propose some principles to address this problem.

In the revision process, a natural requirement is that the operator should follow the principle of minimal of changes [4], which says that a revision should result in a minimal modification. Inspired by the work in [19], we give the definition of fewer change to formalize the above principle.

**Definition 10.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two DL-Lite<sub>FR</sub> TBoxes,  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$  and  $RS = (\mathcal{T}, \mathcal{T}_\mathbb{E}, \mathcal{T}_\mathbb{E}')$  and  $RS' = (\mathcal{T}, \mathcal{T}_\mathbb{E}', \mathcal{T}_\mathbb{E}'')$  be two refinement of revision states w.r.t  $\mathcal{T}$ .  $RS'$  has fewer changes than  $RS$  if

1.  $|\mathcal{T}_\mathbb{E}'| < |\mathcal{T}_\mathbb{E}|$ , or
2.  $|\mathcal{T}_\mathbb{E}'| = |\mathcal{T}_\mathbb{E}|$ , and  $|\mathcal{T}_\mathbb{E}''| < |\mathcal{T}_\mathbb{E}'|$ .

In the above definition,  $RS'$  has fewer changes than  $RS$  means  $RS'$  has fewer unwanted axioms than  $RS$ , or if the number of axioms of  $RS'$  has the same as  $RS$ , then the number of axioms of  $RS'$  should be less than that of  $RS$ .

This definition gives the conditions of selecting axioms when calculating refinement of revision state. In order to satisfy the condition of fewer changes, we give a revision function which is actually a composite of two parts. The first part calculates the unwanted axioms of TBox, and the second part refines valuable information from the results of the first part. The first part of our revision function can be implemented by an incision function [8].

**Definition 11.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two DL-Lite<sub>FR</sub> TBoxes and  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . An incision function for  $\mathcal{T}$ , denoted by  $\sigma$ , is a function  $(\sigma : 2^{2^T} \rightarrow 2^T)$  such that

1.  $\sigma(mips_{\mathcal{T}_2}(\mathcal{T}_1)) \subseteq \bigcup_{\mathcal{T}_i \in mips_{\mathcal{T}_2}(\mathcal{T}_1)} \mathcal{T}_i$ .
2. if  $\mathcal{T}' \in mips_{\mathcal{T}_2}(\mathcal{T}_1)$ , then  $\mathcal{T}' \cap \sigma(mips_{\mathcal{T}_2}(\mathcal{T}_1)) \neq \emptyset$ .

An incision function selects at least one axioms from each non-empty  $MIPS_{\mathcal{T}_2}(\mathcal{T}_1)$ . To meet the conditions of fewer changes, the results should be minimal, that is, no other incision function  $\sigma'$  satisfies  $\sigma'(mips_{\mathcal{T}_2}(\mathcal{T}_1)) \subset \sigma(mips_{\mathcal{T}_2}(\mathcal{T}_1))$ .

In the second part of the revision function, we propose a refinement function which expands the results of incision function  $\sigma$ . We define the refinement function in the following way:

**Definition 12.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two DL-Lite<sub>FR</sub> TBoxes and  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . A refinement function for  $\mathcal{T}$ , denoted by  $\xi$ , is a function  $(\xi : 2^{2^T} \rightarrow 2^T)$  based on incision function  $\sigma$  such that,

1.  $\xi(mips_{\mathcal{T}_2}(\mathcal{T}_1)) \subseteq cl_\mathcal{T}(\phi)$ ,  $\phi \in \sigma(mips_{\mathcal{T}_2}(\mathcal{T}_1))$ .
2.  $\xi(mips_{\mathcal{T}_2}(\mathcal{T}_1)) \cup \mathcal{T}_2$  is coherent.

According to Definitions 11 and 12, we obtain a revision function by composing an incision function and a refinement function, denoted by  $\mathcal{F}_\mathcal{T} = \sigma \oplus \xi$ .

In the composite function  $\mathcal{F}_\mathcal{T} = \sigma \oplus \xi$ , incision function  $\sigma$  selects unwanted axioms from MIPS and refinement function  $\xi$  refines result of  $\sigma$ . However, among different types of revision functions, some of them cannot satisfy the conditions of fewer change. To avoid the above case, we give the following definition.

**Definition 13.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two DL-Lite<sub>FR</sub> TBoxes and  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . A revision function  $\mathcal{F}_\mathcal{T} = \sigma \oplus \xi$  for  $\mathcal{T}$  is optimal with respect to a lexicographic order if there is no other revision function  $\mathcal{F}'_\mathcal{T} = \sigma' \oplus \xi'$  that satisfies the following conditions:

1.  $|\sigma'(mips_{\mathcal{T}_2}(\mathcal{T}_1))| < |\sigma(mips_{\mathcal{T}_2}(\mathcal{T}_1))|$
2. if  $|\sigma'(mips_{\mathcal{T}_2}(\mathcal{T}_1))| = |\sigma(mips_{\mathcal{T}_2}(\mathcal{T}_1))|$ , then  $|\xi'(mips_{\mathcal{T}_2}(\mathcal{T}_1))| < |\xi(mips_{\mathcal{T}_2}(\mathcal{T}_1))|$

Let  $(\mathcal{T}, \mathcal{T}_\mathbb{E} \cup \mathcal{T}_\mathbb{E}', \mathcal{T}_\mathbb{E}'')$  be a refinement revision state of  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . We obtain that  $\mathcal{T}_\mathbb{E} = \sigma(mips_{\mathcal{T}_2}(\mathcal{T}_1))$ ,  $\mathcal{T}_\mathbb{E}' = \xi(mips_{\mathcal{T}_2}(\mathcal{T}_1))$  and  $\mathcal{T}_\mathbb{E}'' = (\mathcal{T} \setminus \mathcal{T}_\mathbb{E})$ .

Based on the above discussion, we give the definition of an operator for revising a TBox  $\mathcal{T}_1$  with a newly received TBox  $\mathcal{T}_2$ . The idea is that we obtain the revision state of the merged TBox by calculating  $mips_{\mathcal{T}_2}(\mathcal{T}_1)$ .

**Definition 14.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two TBoxes and  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . Let  $(\mathcal{T}, \mathcal{T}_\mathbb{E}, \mathcal{T}_\mathbb{E}')$  be a revision state of  $\mathcal{T}$  and  $(\mathcal{T}, \mathcal{T}_\mathbb{E} \cup \mathcal{T}_\mathbb{E}', \mathcal{T}_\mathbb{E}'')$  be the refinement revision state of  $\mathcal{T}$ . The refinement revision operator  $\circ_\mathbb{E}$  is defined as follows:  $\mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2 = (\mathcal{T}_1 \setminus \mathcal{T}_\mathbb{E}') \cup \mathcal{T}_\mathbb{E}' \cup \mathcal{T}_2$ .

In the following, we discuss the properties of the refinement revision operator.

**Proposition 1.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two TBoxes. The operator  $\circ_\mathbb{E}$  satisfies the following properties:

- R1.  $\mathcal{T}_2 \subseteq \mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2$ .
- R2. if  $\mathcal{T}_1 \cup \mathcal{T}_2$  is coherent, then  $\mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2 = \mathcal{T}_1 \cup \mathcal{T}_2$ .
- R3. if  $\mathcal{T}_2$  is coherent, then  $\mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2$  is coherent.
- R4. if  $\mathcal{T}_2 \equiv \mathcal{T}_2'$ , then  $\mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2 \equiv \mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2'$ .
- R5. if  $\phi \in \mathcal{T}_1$  and  $\phi \notin \mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2$ , then there exists  $S_1 \subseteq \mathcal{T}_1$  such that  $S_1 \cup \mathcal{T}_2$  is coherent, but  $S_1 \cup \mathcal{T}_2 \cup \{\phi\}$  is not.

**Proof.** According to Definition 14,  $\mathcal{T}_2 \subseteq (\mathcal{T}_1 \setminus \mathcal{T}_\mathbb{E}') \cup \mathcal{T}_\mathbb{E}' \cup \mathcal{T}_2 = \mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2$ . It implies that (R1) holds. If  $\mathcal{T}_1 \cup \mathcal{T}_2$  is coherent, we can get  $mips_{\mathcal{T}_2}(\mathcal{T}_1) = \emptyset$ , which indicates that  $\mathcal{T}_\mathbb{E}$  and  $\mathcal{T}_\mathbb{E}'$  in a refinement revision state are empty set. It implies that (R2) holds. According to Definition 14, it is clear that (R3) holds. If  $\mathcal{T}_2 \equiv \mathcal{T}_2'$ , we can get  $mips_{\mathcal{T}_2}(\mathcal{T}_1) = mips_{\mathcal{T}_2'}(\mathcal{T}_1)$  according to Definition 7. Thus,  $\sigma(mips_{\mathcal{T}_2}(\mathcal{T}_1)) = \sigma(mips_{\mathcal{T}_2'}(\mathcal{T}_1))$ . It implies that (R4) holds. According to Definition 14, the prerequisite of (R5) indicates that  $\phi \in \mathcal{T}_1 \setminus (\mathcal{T}_1 \circ_\mathbb{E} \mathcal{T}_2) \subseteq \mathcal{T}_\mathbb{E}'$ . In the previous discussion, we know that  $\mathcal{T}_\mathbb{E}'$  is the set of axioms selected by the incision function. Therefore, axiom  $\phi$  satisfied the prerequisite of (R5) must be in a subset of  $\mathcal{T}_1$  which is in conflict with  $\mathcal{T}_2$ . It implies that (R5) holds.  $\square$

#### 4. Graph-based algorithms for TBox debugging and revision

In this section, we present our graph-based algorithms for TBox debugging and revision. In the process of debugging, all MIPPs, which can be transformed to MIPSs of ontology, can be calculated based on  $\mathcal{G}_\mathcal{T}$ . In the process of revision, the incoherence of ontology can be eliminated based on the MIPPs calculated in the process of debugging.

**Algorithm 1:** Graph-based debugging.

**Input:** an incoherent TBox  $\mathcal{T}$

**Output:** set of MIPS in  $G_{\mathcal{T}}$

```

1 Construct  $G_T = \langle N, E \rangle$ ;
2  $mipps \leftarrow \text{CalculateMIPP}(G_T)$  ;
3 for each MIPP in  $mipps$  do
4    $\lfloor mipps \leftarrow mipps \cup \text{TransToAxioms}(MIPP)$ ;
5 return  $mipps$ ;

```

**Algorithm 2:** CalculateMIPP.

**Input:**  $G_{\mathcal{T}}$

**Output:** set of MIPP in  $G_{\mathcal{T}}$

```

1 mipps  $\leftarrow \emptyset$  ;
2 for each  $\neg S \in N$  do
3   for each  $n_1 \in \text{Descendants}(\neg S, G_T)$  do
4     for each  $n_2 \in \text{Descendants}(S, G_T) \cup \{S\}$  do
5       if  $n_1 = n_2$  then
6         pathpairs =
          GetDisjointPathpairs(path( $n_1, \neg S$ ), path( $n_2, S$ ));
7         for each  $\{\langle n_1 \rightarrow \neg S, n_2 \rightarrow S \rangle\}$  in pathpairs do
8           mipps  $\leftarrow \text{mipps} \cup \{\langle n_1 \rightarrow \neg S, n_2 \rightarrow S \rangle\}$ ;
9 return mipps;

```

---

**Algorithm 3:** TransToAxioms.

**Input:** a MIPP in  $G_{\mathcal{T}}$

**Output:** a MIPS in  $\mathcal{T}$

```

1   $MIPS \leftarrow \emptyset$ ,  $axiomset \leftarrow \emptyset$ ;
2  for each  $\langle P_1, P_2 \rangle \in MIPP$  do
3      for  $i = 1 : 2$  do
4          for each  $arc \langle B_1, B_2 \rangle \in P_i$  do
5              if  $B_1 = \exists R_1$  and  $B_2 = \exists R_2$  and  $arc \langle R_1, R_2 \rangle \in G_T$  then
6                   $axiomset \leftarrow axiomset \cup \{R_1 \sqsubseteq R_2\}$ ;
7              else
8                   $axiomset \leftarrow axiomset \cup \{B_1 \sqsubseteq B_2\}$ ;
9   $MIPS \leftarrow axiomset$ ;
10 return  $MIPS$ ;

```

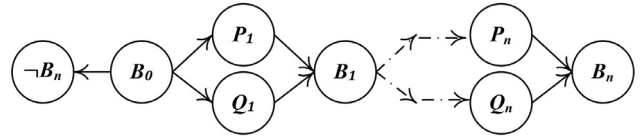
#### 4.1. Graph-based algorithm for TBox debugging

We propose [Algorithm 1](#) for calculating all MIPSs of a TBox. We then give an example in the following to illustrate the procedure of this algorithm.

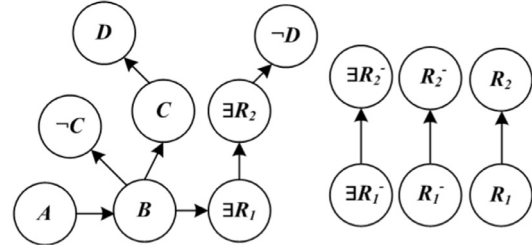
Let  $S$  be a concept or a role expression. At the beginning of [Algorithm 1](#), the TBox will be transformed into a graph in line 1, and then [Algorithm 2](#) will be invoked to calculate MIPPs. In lines 2–6 of [Algorithm 2](#), descendant nodes of  $S$  and  $\neg S$  can be obtained. If there is a node (called  $S'$ ) between descendant nodes of node  $S$  (specially, it includes  $S$  itself for the case  $S \sqsubseteq \neg S$ ) and those of node  $\neg S$ . This also means that there exists a path-pair  $\langle S' \rightarrow S, S' \rightarrow \neg S \rangle$  on  $G_T$ . Since  $S' \rightarrow S$  and  $S' \rightarrow \neg S$  have no joint edge,  $\langle S' \rightarrow S, S' \rightarrow \neg S \rangle$  is a MIPP.

The function *GetDisjointPathpairs* is applied to calculate path-pairs between two disjoint nodes, and there is no intersection edge between these path-pairs. Algorithm 3 is applied to transform a MIPP into a MIPS.

In the following, we first discuss the complexity of Algorithm 1–3. According to the construction rules of graph,



**Fig. 1.** Directed graph in Example 4.



**Fig. 2.** Directed graph in Example 5.

line 1 of [Algorithm 1](#) can be executed in polynomial time. In line 2 of [Algorithm 1](#), [Algorithm 2](#) (*CalculateMIPP*) is invoked. Thus, we turn to discuss the complexity of [Algorithm 2](#).

In line 2 of [Algorithm 2](#), all the negative nodes of  $G_T$  can be accessed in the first loop. In lines 3–4 of [Algorithm 2](#), the descendant nodes of each negative node  $\neg A$  and its disjoint node  $A$  can be calculated. At the end of [Algorithm 2](#), all of MIPP can be obtained in lines 7–8 which is nested in the third loop. From above analysis, [Algorithm 2](#) is actually executed in a quadruple loop, but the complexity of function *GetDisjointPathPairs* which is invoked in line 6 of [Algorithm 2](#) is in NP-hard. *GetDisjointPathPairs* aims to calculate all pairs of paths between the nodes  $\neg A$  and  $A$ . Calculating a pair of paths is a traversal of the graph  $G_T$  to visit all nodes and edges in the graph, thus its complexity is  $O(|V|^2)$  in the worst case. Since there exists exponential number of pairs of paths in the worst case, there may be exponentially MIPP in the  $G_T$ . It can be shown in the following example which is modified from [Example 1](#) of [\[20\]](#).

**Example 4.** Given a TBox  $\mathcal{T}$ , where  $\mathcal{T} = \{B_0 \sqsubseteq \neg B_n, B_{i-1} \sqsubseteq P_i, B_{i-1} \sqsubseteq Q_i, P_i \sqsubseteq B_i, Q_i \sqsubseteq B_i \mid 1 \leq i \leq n\}$ .

The TBox  $\mathcal{T}$  can be transformed into a graph as shown in Fig. 1. In Fig. 1, we can obtain that the number of pairs of paths from  $B_0 \rightarrow \neg B_n$  and  $B_0 \rightarrow B_n$  is  $2^n$ . Therefore, the complexity of Algorithm 2 is in exponential time.

We continue to discuss Algorithm 1. Lines 3–4 of Algorithm 1 contain a single loop which can be executed in time  $O(n)$  ( $n$  is the size of `mipps`). From what we have discussed above, the complexity of Algorithm 1 is also in exponential time.

In Algorithm 3, a MIPP can be transformed into a MIPS. Lines 2–8 of Algorithm 3 contain a triplet loop which can be executed in cubic time. Therefore, the complexity of Algorithm 3 is  $O(n^3)$ .

**Example 5.** Given a TBox  $\mathcal{T}$ , where  $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C, C \sqsubseteq D, B \sqsubseteq \neg C, R_1 \sqsubseteq R_2, B \sqsubseteq \exists R_1, \exists R_2 \sqsubseteq \neg D\}$ .

According to the Construction Rules, we construct a directed graph as shown in Fig. 2. In lines 2–6 of Algorithm 2, two MIPPs are identified in Fig. 2,  $\{B \rightarrow \neg C, B \rightarrow C\}$  and  $\{B \rightarrow D, B \rightarrow \neg D\}$ . In Algorithm 3, the path pair of MIPP will be transformed to axioms. For  $\exists R_1 \sqsubseteq \exists R_2 \not\sqsubseteq T$  and  $R_1 \models \exists R_2 \sqsubseteq \exists R_2$ ,  $\text{arc}(\exists R_1, \exists R_2)$  corresponds to  $R_1 \sqsubseteq R_2$  in  $T$ . Finally, two MIPSs  $\{B \sqsubseteq C, B \sqsubseteq \neg C\}$  and  $\{B \sqsubseteq D, C \sqsubseteq D, R_1 \sqsubseteq R_2, B \sqsubseteq \exists R_1, \exists R_2 \sqsubseteq \neg D\}$  are generated.

**Theorem 9.** Given an incoherent TBox  $\mathcal{T}$ , the output of Algorithm 1 is the set of all MIPSs of  $\mathcal{T}$ .

**Proof.** According to Theorem 5,  $n \in \text{descendants}(S, G_{\mathcal{T}})$  means  $n \sqsubseteq S$ . Thus, in lines 2–6 of Algorithm 2,  $n_1 = n_2$  means that there

exists a path pair from the node  $n_1$  (or  $n_2$ ) to disjointness pair  $(S, \neg S)$ . By Definition 6, these two paths are in a MIPP. Further, Algorithm 3 is applied to transform a MIPP to a MIPS.

For any MIPS in  $\mathcal{T}$ , there exists a concept (or role) that is subsumed by two disjointness concepts (or roles). Thus, from Theorem 5, we can get two intersectional paths derived from two disjointness nodes in  $G_{\mathcal{T}}$ . Obviously, the intersectional point is the descendant of the two disjointness node. Namely, each MIPS can be calculated as shown in lines 1–4 of Algorithm 1. Therefore, the result of Algorithm 1 is  $mips(\mathcal{T})$ .  $\square$

#### 4.2. Graph-based algorithms for TBox revision

In this subsection, we give specific algorithms to instantiate the refinement revision operator defined in Section 3. Inspired by the works reported in [7,8] and [21], we propose two algorithms, one is based on a scoring function, the other is based on a hitting set tree (HS-Tree).

##### 4.2.1. Revision algorithm based on scoring function

In [22], a scoring function is used to measure the inconsistency of an ontology. In [8], this scoring function is reformulated for ontology revision in which axioms with the highest frequency of occurrence in  $mips_{\mathcal{T}_2}(\mathcal{T}_1)$  are picked. In the following, we introduce the definition of a scoring function.

**Definition 15.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two DL-Lite<sub>FR</sub> TBoxes and  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . The scoring function for axiom  $\alpha$  w.r.t  $\mathcal{T}$ , is a function  $S_{\mathcal{T}} : mips_{\mathcal{T}_2}(\mathcal{T}_1) \mapsto N$  such that for all  $\phi \in \bigcup_{mips} S_{\mathcal{T}}(\phi) = |\mathcal{T}_1 \in mips_{\mathcal{T}_2}(\mathcal{T}_1) : \phi \cap \mathcal{T}_1 \neq \emptyset|$ .

Then, in order to define a specific refinement function, we modify the notion of axiom impact in [18], which evaluates the impact of an axiom when removing it from TBox.

**Definition 16.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two DL-Lite<sub>FR</sub> TBoxes and  $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ . The impact of an axiom  $\phi$ , denoted by  $impact(\phi)$ , is defined as follows:

$$impact(\phi) = \begin{cases} 0, & \phi \notin \bigcup_{mips} \\ |\{\phi' \in cl_{\mathcal{T}}(\phi) \mid \bigcup_{mips_{\phi}} \setminus \{\phi\} \cup \{\phi'\} \text{ is coherent}\}|, & \phi \in \bigcup_{mips} \end{cases}$$

In Definition 16,  $\bigcup_{mips} = \bigcup_{\mathcal{T}_1 \in mips_{\mathcal{T}_2}(\mathcal{T}_1)} \mathcal{T}_1$ ,  $mips_{\phi}$  denotes the set of MIPS that contains the inclusion assertion  $\phi$  and  $\bigcup_{mips_{\phi}} = \bigcup_{\mathcal{T}_1 \in mips_{\phi}} \mathcal{T}_1$ .

Based on the value of the scoring function, the incision function can select a candidate axiom  $\phi$ . In detail, an axiom will be selected if its score is maximum, otherwise, the impact value of these axioms will be calculated. Then, the axiom with minimum value of impact will be selected because the smaller the value of impact, the less number of axioms will be added.

By using the scoring function and the axiom impact, we present algorithm *GReviByScoring* (see Algorithm 4) for TBox revision, which consists of three steps.

**Step 1 initialization.** In lines 1–2, the set  $E_1$  (resp.  $E_2$ ) is initially assigned by the set of edges of  $G_{\mathcal{T}_1}$  (resp.  $G_{\mathcal{T}_2}$ ). *CalculateMIPP* is invoked to obtain the set of MIPP and the returned result is assigned to *mipp* (line 3), and then the edges of *mipp*, which belong to  $G_{\mathcal{T}_1}$ , are selected to initialize the set *candidates* (lines 4–7). At the end of the step, the function *SortingOnScore* sorts *candidates* according to the frequency of the edges in *mipp*.

**Step 2 calculating the unwanted set.** In lines 9–15, the edge with maximum score is selected. The function *GetTopEqEdges* (line 13) groups the edges with the same maximum score in *eqEdges*. The function *GetImpact* is called to rank the impact values of the edges in *eqEdge* (line 14). The edge with minimum impact value is then

#### Algorithm 4: GReviByScoring.

---

**Input:**  $G_{\mathcal{T}} = G_{\mathcal{T}_1} \cup G_{\mathcal{T}_2}$   
**Output:**  $G_{\mathcal{T}_1} \circ_{\neq} G_{\mathcal{T}_2}$

```

1  $E_{\neq} \leftarrow \emptyset, E_{\neq} \leftarrow \emptyset, E_{\neq}^r \leftarrow \emptyset;$ 
2  $E_1 \leftarrow GetEdges(G_{\mathcal{T}_1}), E_2 \leftarrow GetEdges(G_{\mathcal{T}_2});$ 
3  $mipp = CalculateMIPP(G_{\mathcal{T}});$ 
4 for each  $MIPP \in mipp$  do
5   for each  $edge \in MIPP$  do
6     if  $edge \in G_{\mathcal{T}_1}$  then
7        $candidates \leftarrow candidates \cup \{edge\};$ 
8  $scoredEdges \leftarrow SortingOnScore(candidates);$ 
9 while  $mipp \neq \emptyset$  do
10  if  $ExistMaxScore(scoredEdges)$  then
11     $edge \leftarrow GetMaxScoreEdge(scored);$ 
12  else
13     $eqEdges \leftarrow GetTopEqEdges(scoredEdges);$ 
14     $rankedEdges \leftarrow GetImpact(eqEdges, G_{\mathcal{T}});$ 
15     $edge \leftarrow GetMaxRankEdge(rankedEdges);$ 
16   $E_{\neq} \leftarrow E_{\neq} \cup \{edge\};$ 
17   $rCandidates \leftarrow rCandidates \cup Closure(edge, G_{\mathcal{T}});$ 
18   $UpdateState(edge, scoredEdges, mipp);$ 
19  $E_{\neq} = (E_1 \setminus E_{\neq}) \cup E_2;$ 
20 for each  $edge \in rCandidates$  do
21  if not  $HasMIPP(E_{\neq} \cup edge)$  then
22     $E_{\neq}^r \leftarrow E_{\neq}^r \cup \{edge\};$ 
23 return  $E_{\neq} \cup E_{\neq}^r;$ 
```

---

selected and put in the unwanted set  $\mathcal{T}_{\neq}$  (line 15). In line 17, the function *Closure* is called to compute the closure of  $G_{\mathcal{T}}$ , and the output is *rCandidates*. Finally, the function *UpdateState* removes the MIPPs that contain the selected edges.

**Step 3 calculating the refinement set.** The set  $E_{\neq}$  is first assigned with  $(E_1 \setminus E_{\neq}) \cup E_2$  (line 19). *rCandidates* is then scanned and checked to see whether a union of an edge in *rCandidates* with  $E_{\neq}$  leads to a new MIPP. All the edges that would not lead to new MIPPs are added in the refinement set  $E_{\neq}$  (lines 20–22).

In the following, we discuss the complexity of Algorithm 4. The exponential algorithm *CalculateMIPP* will be invoked in line 3 for calculating all the MIPP of  $G_{\mathcal{T}}$ . In lines 4–7, there exists a double loop. The times of the outer loop is the number of MIPPs and the times of the inner loop is the number of edges in corresponding MIPP. In line 8, a sorting algorithm will be invoked and its complexity is  $O(n^2)$  in the worst case. The function of line 9–18 is to calculate the unwanted set, which can be executed in polynomial time. Therefore, except for the calculation of MIPP, the complexity of Algorithm 4 is in PTIME.

**Example 6.** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be two TBoxes.  $\mathcal{T}_1 = \{A \sqsubseteq B, B \sqsubseteq C, D \sqsubseteq E, D \sqsubseteq F, F \sqsubseteq G\}$ ,  $\mathcal{T}_2 = \{A \sqsubseteq \neg C, A \sqsubseteq \neg B, C \sqsubseteq E, F \sqsubseteq \neg E\}$ . We consider using algorithm *GReviByScoring* to revise  $\mathcal{T}$ .

We firstly encode  $\mathcal{T}_1$  and  $\mathcal{T}_2$  into the graph as shown in the left part of Fig. 3.

According to Definition 6, we can obtain three MIPPs as follows:  $\{A \rightarrow \neg B, A \rightarrow B\}$ ,  $\{A \rightarrow B, B \rightarrow C, A \rightarrow \neg C\}$ ,  $\{D \rightarrow E, D \rightarrow F, F \rightarrow \neg E\}$ . Based on lines 4–7, we obtain *candidates* =  $\{A \rightarrow B, B \rightarrow C, D \rightarrow E, D \rightarrow F\}$ , whose edges are from  $G_{\mathcal{T}_1}$ .

In line 8, the edges of *candidates* are sorted by the scoring function (see  $scoredEdges = \{(A \rightarrow B, 2), (B \rightarrow C, 1), (D \rightarrow E, 1), (D \rightarrow F, 1)\}$ ).

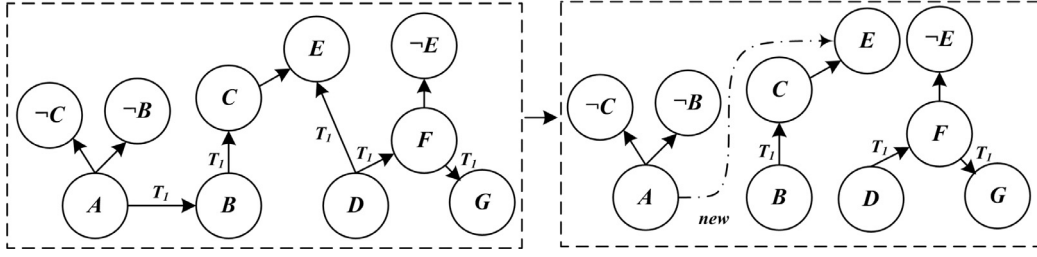


Fig. 3. The process of revision in Example 6.

At the first iteration of lines 9–15, the edge  $(A \rightarrow B)$  is selected in lines 10–11, which has the maximum value of scoring function. Then, edge  $(A \rightarrow B)$  is appended in the set  $E_{\#}$  and the closure of  $(A \rightarrow B)$ , which is  $\{A \rightarrow C, A \rightarrow E\}$ , is appended into the set  $nCandidates$ . Then, the variables in the algorithm are updated. During the updating, the edge  $(A \rightarrow B)$  is removed from  $candidates$ . The MIPPs including the edge  $(A \rightarrow B)$  are removed from  $mipp$  and the  $scoredEdges$  is updated into  $\{(D \rightarrow E, 1), (D \rightarrow F, 1)\}$ . At the second iteration, there are two edges in  $candidates$  with the same value of scoring function. Therefore, the algorithm switches to “else case”, where the impact values are calculated for the edges,  $(D \rightarrow E)$  and  $(D \rightarrow F)$ , based on lines 13–15. For the example, we have obtained that  $Closure(D \rightarrow E, G_T) = \emptyset$  and  $Closure(D \rightarrow F, G_T) = \{D \rightarrow G, D \rightarrow \neg E\}$ . Further, the edge  $(D \rightarrow F)$  is selected based on the function  $GetMaxRankEdge$ . Finally, when  $mipp$  turns into an empty set, the whole iteration finishes.

The above operation just gets unwanted set of edges from  $G_T$ , there still has some available information in set  $nCandidates$ . Thus, in lines 20–22, we retrieve the set  $nCandidates$  and collect the edges in  $nCandidates$  that will not lead to new MIPP. For the example, edge  $(A \rightarrow C)$  is collected. The final result of the revision of Example 6 is shown in the right part of Fig. 3.

#### 4.2.2. Revision algorithm based on hitting set tree

The incision function can be also implemented by a HS-Tree. HS-Tree is proposed in [23] where Reiter propose an approach that constructs a labeled tree based on a conflict set and reformulated in [7] for ontology debugging. During the process of ontology debugging,  $mips_{T_2}(T_1)$  can be regarded as a set of conflict sets. In a HS-Tree, nodes can be labeled with  $MIPS \in mips_{T_2}(T_1)$  and edges can be labeled with axioms  $\alpha \in \bigcup_{T \in mips_{T_2}(T_1)} T$ .

In the following, we propose the revision algorithm, *GReviByHST* (see Algorithm 5).

The algorithm also has three steps which are analogous to *GReviByScoring*. The main difference is in Step 2. In Step 2 of *GReviByHST*, the whole unwanted set  $E_{\#}$  is obtained based on the method of hitting set tree, and the *candidate* set of unwanted axioms are calculated once by invoking the function *HS-Tree* (line 8) and *GetMinHS*.

Next, we discuss the complexity of Algorithm 5. Algorithm 5 aims to perform the revision of ontologies by using a global optimal solution, which firstly calculates all the MIPP of  $G_T$  resembling to Algorithm 4. In line 3, the exponential algorithm *CalculateMIPP* will be invoked. Lines 4–7 contain a double loop which can be executed in square time. In line 8, a function based on hitting set tree will be invoked. There exists exponential number of hitting sets [24] which can be executed in exponential time. There is no loop statement in lines 9–16 which can be executed in polynomial time. Lines 17–19 contain a single loop which can be done in  $O(n)$  ( $n$  is the size of  $rCandidates$ ). Therefore, except for the calculating of MIPP, the complexity of Algorithm 5 is also in exponential time.

#### Algorithm 5: GReviByHST.

---

**Input:**  $G_T = G_{T_1} \cup G_{T_2}$   
**Output:**  $G_{T_1} \circ_{\#} G_{T_2}$

```

1  $E_1 \leftarrow GetEdges(G_{T_1}), E_2 \leftarrow GetEdges(G_{T_2});$ 
2  $E_{\#} \leftarrow \emptyset, E_{\#} \leftarrow \emptyset, E_{\#}^r \leftarrow \emptyset;$ 
3  $mipp = CalculateMIPP(G_T);$ 
4 for each MIPP  $\in mipp$  do
5   for each edge  $\in MIPP$  do
6     if edge  $\in G_{T_2}$  then
7       MIPP  $\leftarrow MIPP \setminus \{edge\};$ 
8  $hittingSets \leftarrow HS-Tree(mipp);$ 
9 if ExistMinHS( $hittingSets$ ) then
10    $E_{\#} \leftarrow GetMinHS(hittingSets);$ 
11 else
12    $eqHSs \leftarrow GetTopEqHSs(hittingSets);$ 
13    $rankedHSs \leftarrow GetImpacts(eqHSs, G_T);$ 
14    $E_{\#} \leftarrow GetMaxRankHS(rankedHSs);$ 
15  $rCandidates \leftarrow Closure(E_{\#}, G_T);$ 
16  $E_{\#} = (E_1 \setminus E_{\#}) \cup E_2;$ 
17 for each edge  $\in rCandidates$  do
18   if not HasMIPP( $E_{\#} \cup edge$ ) then
19      $E_{\#}^r \leftarrow E_{\#}^r \cup \{edge\};$ 
20 return  $E_{\#} \cup E_{\#}^r$ 
```

---

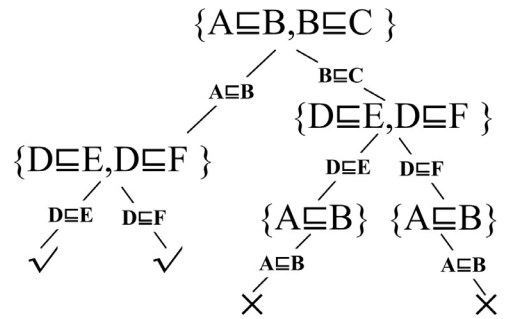


Fig. 4. Hitting set tree of Example 7.

**Example 7.** We consider using the TBoxes in Example 6 here. We perform the revision based on *GReviByHST*.

*GReviByHST* firstly initializes the variables. According to lines 3–7, we get three  $MIPS_{T_2}(T_1)$ :  $\{A \rightarrow B\}$ ,  $\{A \rightarrow B, B \rightarrow C\}$ ,  $\{D \rightarrow E, D \rightarrow F\}$ . According to the algorithm for calculating diagnoses using hitting set [7], we can build a hitting set tree, as shown in Fig. 4.

From Fig. 4, we obtain two minimum hitting sets:  $\{A \rightarrow B, D \rightarrow E\}$  and  $\{A \rightarrow B, D \rightarrow F\}$ . According to lines 9–14, function *GetImpacts* is called to calculate the impact value of these two sets. For  $\{A \rightarrow B, D \rightarrow E\}$ , the refinement set is  $\{A \rightarrow C, A \rightarrow E, D \rightarrow \neg E\}$ . For



**Table 1**  
Partitioned ontologies.

Ontology		Axioms	NIs	Concepts	Roles
AEO	Part 1	1043	1	225	7
	Part 2	1038	4	223	6
CL	Part 1	12373	19	2841	26
	Part 2	12365	20	2843	22
DOLCE	Part 1	294	5	34	66
	Part 2	335	13	33	70
Economy	Part 1	1726	34	285	41
	Part 2	1719	37	296	38
Fly-Anatomy	Part 1	16766	31	7091	27
	Part 2	16703	30	7093	31
FMA	Part 1	274501	2	70339	15
	Part 2	272731	1	69968	14
GO	Part 1	50774	1	17209	1
	Part 2	51091	2	17302	1
Plant	Part 1	10984	24	1464	10
	Part 2	11029	2	1457	9
Terrorism	Part 1	1038	1	93	120
	Part 2	984	1	77	111
Transportation	Part 1	1415	158	387	79
	Part 2	1420	159	383	76

**Table 2**  
Conflict information in ontologies.

	AEO	CL	DOL.	Eco.	F.A.	FMA	GO	Pla.	Terr.	Tra.
UC	49	59	33	51	71	304	97	45	14	62
MIPS	17	25	5	47	53	3	18	12	5	36

$\{A \rightarrow B, D \rightarrow E\}$ , the refinement set is  $\{A \rightarrow C, A \rightarrow E\}$ . In order to guarantee minimal change, we choose the set  $\{A \rightarrow B, D \rightarrow E\}$  as the unwanted set. Based on lines 17–19, the edges in  $rCandidates$  are added.

## 5. Implementation and evaluation

We have implemented the debugging and revision algorithms presented in the previous section. The system is built on several open source software packages. We parse the structure of an ontology by OWLAPI,<sup>1</sup> which is a tool for managing OWL ontologies [25]. We transform an ontology into a graph and store it in the Neo4j graph database [26]. Neo4j<sup>2</sup> is an open-source and high-performance graph database supported by Neo Technology [27]. The main ingredients of Neo4j are nodes and relationships. The nodes of Neo4j are used to record data in nodes which have user-defined properties and organized by relationships which also have user-defined properties. Furthermore, Neo4j provides a powerful declarative graph query language, cypher, which can be leveraged to querying and updating a graph database.

The ontologies used in our experiments have different sizes and structures. An ontology will be approximated [16], if it cannot be expressed in DL-Lite. As we mentioned above, the incoherence appears in the merging process of two ontologies. To obtain the test ontologies, we partition an ontology into two different parts and modify them by inserting some “incoherent-generating” axioms randomly, such as negative inclusions. For example, if  $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C, B \sqsubseteq D\}$ , it is obvious that  $\mathcal{T}$  is coherent. To make it incoherent, we may insert  $C \sqsubseteq \neg D$  into  $\mathcal{T}$ .

In our work, we assume that the single ontology is consistent and the incoherence appears when ontologies are combined. In the preprocessing, we store the two ontologies into the Neo4j database, and use cypher query of neo4j to guarantee the precon-

**Table 3**  
Time (in milliseconds) required to calculate all MIPS.

Ontology	Pellet	Hermit	FaCT++	JFaCT	onGraph
AEO	1385	1986	814	1602	397
CL	1536	2423	1187	2342	652
DOL.	784	1121	515	867	362
Eco.	1354	1429	677	1976	652
F.A.	2129	3386	1926	4993	642
FMA	47975	49867	45760	92017	3718
GO	3805	5442	3893	9465	906
Pla.	1654	2180	1102	1978	596
Terr.	415	496	405	521	193
Tra.	1892	2861	1457	4926	725

**Table 4**  
Debugging time (in milliseconds) for incremental unsatisfiable concepts.

UC in GO	Pellet	Hermit	FaCT++	JFact	onGraph
100	4381	6090	3991	8882	827
110	4641	6280	4342	10864	912
120	4982	7155	4948	12636	933
130	6609	9771	6425	17174	1016
140	8077	10132	7305	66663	1127
150	9449	12082	8714	overflow	1209
160	10210	13065	9630	overflow	1263
200	12889	16573	12503	overflow	1769

**Table 5**  
Time (in seconds) required to ontology revision.

	AEO	CL	DOL.	Eco.	F.A.
<i>GReviByScoring</i>	14.076	16.649	2.358	20.198	20.452
<i>GReviByHST</i>	21.972	timeout	4.776	timeout	timeout
	FMA	GO	Pla.	Terr.	Tra.
<i>GReviByScoring</i>	4.524	14.448	4.879	2.644	18.393
<i>GReviByHST</i>	6.47	250.552	8.53	5.179	timeout

dition of our algorithms when adding a disjoint axiom to an ontology. The statistics of the used ontologies is given in Table 1.

All experiments are performed on a desktop computer with Intel Core i5-2400 3.1 GHz CPU and 8GB of RAM, running Microsoft window 7 operating system, and Java 1.7 with 6GB of heap space.

### 5.1. Experiment of computing MIPSs based on graph

In this paper, our approach of debugging aims to calculate MIPS. Many ontology reasoners can calculate MIPS. In this section, we performed comparative experiments with several popular reasoners, such as Pellet,<sup>3</sup> Hermit,<sup>4</sup> FaCT++,<sup>5</sup> JFaCT<sup>6</sup>.

We conducted the experiment from two aspects: one is to compare efficiency and the other is to compare performance on an ontology with different number of unsatisfiable concepts or roles.

The conflicting information is shown in Table 2 (we abbreviate the name of ontologies in Table 1 and UC is the abbreviation of unsatisfiable concepts). The experimental results are shown in Table 3. From Table 3 (Graph-based Debugging approach is abbreviated as onGraph), our approach outperforms others. Especially, for some complex ontologies, such as FMA, our approach is significantly faster than others. Table 4 gives the result of scalability over the adapted GO ontology, with different number of unsatisfiable concepts or roles added. In the table, overflow means that the experimental program exhausts available memory.

<sup>3</sup> <https://github.com/complextible/pellet>.

<sup>4</sup> <http://hermit-reasoner.com>.

<sup>5</sup> <https://code.google.com/p/factplusplus>.

<sup>6</sup> <http://jfact.sourceforge.net>.

<sup>1</sup> <http://owlapi.sourceforge.net>.

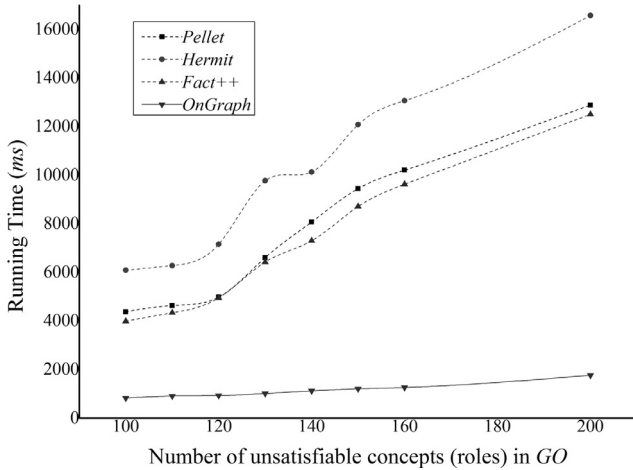
<sup>2</sup> <http://www.neo4j.org/learn/cypher>.

**Table 6**  
The change of ontologies in *GReviByScoring*.

	AEO	CL	DOL	Eco.	F.A.	FMA	GO	Pla.	Terr.	Tra.
Deleted	6	12	3	27	3	1	7	3	5	14
Added	6	22	6	144	35	1	13	3	5	66

**Table 7**  
The change of ontologies in *GReviByHST*.

	AEO	CL	DOL	Eco.	F.A.	FMA	GO	Pla.	Terr.	Tra.
Deleted	5	–	3	–	–	1	7	3	5	–
Added	5	–	6	–	–	1	13	3	5	–



**Fig. 5.** Time required to calculate MIPSs on GO.

We first identified all the disjoint concepts in GO ontology, and then increased the number of unsatisfiable concepts by adding common subsumed concept to the disjoint concepts randomly (we began with 100 unsatisfiable concepts for clearer comparison). The same treatment is also applied to roles. To have a better view, we transformed Table 4 into chart shown in Fig. 5. From Fig. 5, we can see that, with the growing number of unsatisfiable concepts and roles, the execution time of the state of the art approaches increases sharply. In comparison, the execution time of our approach grows moderately. The comparison result shows that our approach performs more steadily than other approaches in computing MIPS.

### 5.2. Experiment of revising terminologies based on graph

We conducted the experiment of revision to compare *GReviByScoring* and *GReviByHST*.

In this part, we set 30 min as the cut-off time. The experimental result of TBox revision is shown in Table 5. From Table 5, we can see that scoring-based algorithm outperformed HST-based algorithm. We can further see that, with the number of MIPP increasing, the execution time is also increasing and *GReviByHST* runs slower than *GReviByScoring*.

The difference between the two algorithms is caused by the specific incision function. *GReviByHST* is based on the method of hitting set tree to implement the incision function that can obtain the globally optimal solution. However, the complexity of finding minimal hitting sets is known to be NP-hard [24]. Thus, *GReviByHST* leads to a non-linear growth of running time. *GReviByScoring* employs a scoring function to define the incision function. Though it only achieves locally optimal solution, the complexity of calculating cardinality-minimal set is in PTime. Hence, the execution time of *GReviByScoring* is more than that of *GReviByScoring*.

The changes of the number of axioms in ontologies are collected in Tables 6 and 7. As shown in Tables 6 and 7, only on the ontology AEO, *GReviByHST* leads to fewer changes than that of *GReviByScoring*.

## 6. Related work

In the process of building and maintaining ontologies, new information may be inconsistent with ontologies when added it to the ontologies. Reasoning with inconsistent ontologies can obtain arbitrary conclusions. Therefore, inconsistency of ontology is an urgent problem that has to be resolved. This problem can be tackled by ontology debugging and revision which have been widely discussed in the literature.

First, we introduce the related work of ontology debugging. In general, the approaches for ontology debugging can be divided into two categories: glass-box approach and black-box approach. Glass-box approaches are based on reasoning algorithms and most of the glass-box approaches are obtained as extension of tableau-based algorithms for checking satisfiability of a DL-based ontology. The first tableau-based algorithm for ontology debugging is proposed in [14], but the algorithm is restricted to unfoldable  $\mathcal{ALC}$  TBoxes. In [5], the tableau-based algorithm is extended to a more expressive DL, OWL DL. As pointed out in [21], one problem for the tableau-based algorithms is that some blocking techniques cannot be used. To avoid this problem, an automata-based approach is given in [28]. This approach has some nice theoretical advantage over tableau-based approaches. In order to compute MUPS, a debugging algorithm for DL-Lite ontologies is proposed based on the  $cln(T)$  which can be calculated off-line [29]. Our graph-based debugging algorithm belongs to the glass-box approach and specifically designed for DL-Lite ontologies. Furthermore, our algorithm can directly compute MIPS, while previous debugging algorithms are mostly applied to compute MUPS.

Different from the glass-box approaches, the black-box approaches do not need to modify the DL reasoner. However, in the worst case, the black-box approach may call the reasoner an exponential number of times. In [30], a general DL-based debugging approach is given. The approach is based on Reiter's algorithm and employs a simplified variant of QUICKPLAIN[31] to generate the hitting set tree. According to [20], computing MUPS is a hard task for tractable DLs like  $\mathcal{EL}$ . Therefore, it may be not possible to compute all MUPS for some large ontologies. In [32], the authors present an extensive evaluation of existing ontology debugging systems. However, they mainly evaluate systems for computing MUPS, but do not consider the computation of MIPS.

Next, we discuss the related work of ontology revision. Ontology revision deals with the incoherent problem when incorporating newly received information. In a nutshell, approaches to ontology revision can roughly be divided into two categories: syntax-based one and model-based one.

Most of practical revision operators are syntax dependent, i.e., if two logically dependent ontologies are revised by another ontology, the results of revision may not be logically equivalent. One type of syntax-based approaches is based on incision function. The incision function is used to select axioms, which are to be removed from the original ontology. The authors in [8] proposes a kernel revision operator and give two algorithms to define specific kernel revision operators. The authors in [9] develop a revision operator based on a trust-based incision function. The other type of formula-based approaches is to find maximally satisfiable terminologies. An algorithm is proposed in [33] to find maximally satisfiable terminologies for the description logic  $\mathcal{ALC}$ . A fine-grained approach is proposed in [34] in which notion of fine-grained repair is firstly introduced. In the approach, one axiom may be weakened to more axioms instead of being deleted.

Most of the works on model-based revision in DLs are devoted to proving the inexpressibility of model-based revision operators (see [35] and [36] for example). However, very few of them discuss the approximation of model-based revision. One exception is the work in [37], where a revision operator is defined by a new semantics called features. However, feature-based revision also suffers from the inexpressibility problem and the algorithm is inefficient to deal with large ABoxes. Recently, there are some works on TBox revision based on a new semantics, called type semantics (see [38] and [39]). In [40], the authors propose an approach for approximating model-based revision operator by using a syntax-based revision operator. However, this work focuses on ABox revision instead of TBox revision.

## 7. Conclusion

In this paper, we proposed graph-based approaches for debugging and revising incoherent DL-Lite ontologies.

In order to debug incoherent ontologies, we firstly encoded a DL-Lite ontology into a directed graph according to the given construction rules. Then, we gave a definition of MIPP on the graph and proposed the debugging approach based on MIPP. Finally, we conducted experiments over some adapted ontologies. Without the help of the DL reasoner, all the MIPS of an ontology can be calculated by backtracking some pairs of nodes in the graph. Our graph-based approach improves the ontology debugging both in efficiency and scalability.

In order to revise incoherent ontology, we propose the refinement revision operation based on a refinement revision space which can be obtained by a revision function. According to the principle of minimal change, the refinement revision operation can reserve more valid information. Finally, we gave two algorithms for ontology revision. one is based on incision function and the other is based on a HS-Tree. The experimental results shown that scoring-based algorithm leads to a better performance than HST-based algorithm, especially for the ontologies that contain numerous MIPSs.

In the paper, we focused on the family of DL-Lite. One of future works, we plan to extend our algorithms to more expressive description languages. As to DL-Lite, one of the remarkable characteristics is that it is suitable for query answering. The data complexity of answering unions of conjunctive queries in DL-Lite is polynomial in the size of the whole ontology and in LOGSPACE with respect to the size of ABox only [11]. As another of future work, we plan to study consistent query answering approaches based on graph.

## Acknowledgments

We would like to thank the anonymous referees for their comments. Research presented in this paper was partially supported by National Science Foundation of China (no. 61272378), by the funds from JiangXi Educational Committee (no. GJJ12643), by the National High Technology Research Program of China (863 Program) (no. 2015AA015406).

## References

- [1] K. Amaief, J. Lu, Ontology-supported case-based reasoning approach for intelligent m-government emergency response services, *Decis. Support Syst.* 55 (1) (2013) 79–97.
- [2] Y. Ma, B. Jin, Y. Feng, Dynamic evolutions based on ontologies, *Knowl. Based Syst.* 20 (1) (2007) 98–109.
- [3] B. Parsia, E. Sirin, A. Kalyanpur, Debugging OWL ontologies, in: *Proceedings of the 14th International Conference on World Wide Web (WWW)*, 2005, pp. 633–640.
- [4] G. Qi, F. Yang, A survey of revision approaches in description logics, in: *Proceedings of Second International Conference on Web Reasoning and Rule Systems (RR)*, 2008, pp. 74–88.
- [5] A. Kalyanpur, B. Parsia, E. Sirin, J. Hendler, Debugging unsatisfiable classes in OWL ontologies, *J. Web Sem.* 3 (4) (2005) 268–293.
- [6] Q. Ji, Z. Gao, Z. Huang, M. Zhu, An efficient approach to debugging ontologies based on patterns, in: *Proceedings of Joint International Semantic Technology Conference (JIST)*, 2011, pp. 425–433.
- [7] S. Schlobach, Z. Huang, R. Cornet, F. Van Harmelen, Debugging incoherent terminologies, *J. Autom. Reasoning* 39 (3) (2007) 317–349.
- [8] G. Qi, P. Haase, Z. Huang, J.Z. Pan, A kernel revision operator for terminologies, in: *Proceedings of the 21st International Workshop on Description Logics (DL)*, 2008, pp. 419–434.
- [9] J. Golbeck, C. Halaschek-Wiener, Trust-based revision for expressive web syndication, *J. Log. Comput.* 19 (5) (2009) 771–790.
- [10] C.E. Alchourrón, P. Gärdenfors, D. Makinson, On the logic of theory change: partial meet contraction and revision functions, *J. Symb. Log.* 50 (2) (1985) 510–530.
- [11] D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The DL-Lite family, *J. Autom. Reasoning* 39 (3) (2007) 385–429.
- [12] T. Venetis, G. Stoilos, G.B. Stamou, Query rewriting under query refinements, *Knowl. Based Syst.* 56 (2014) 36–48.
- [13] A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyashev, The DL-Lite family and relations, *J. Artif. Intell. Res.* 36 (1) (2009) 1–69.
- [14] S. Schlobach, R. Cornet, Non-standard reasoning services for the debugging of description logic terminologies, in: *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 355–362.
- [15] S. Gao, G. Qi, H. Wang, A new operator for ABox revision in DL-Lite, in: *Proceedings of the 26th National Conference on Artificial Intelligence (AAAI)*, 2012, pp. 2423–2426.
- [16] D. Lembo, V. Santarelli, D.F. Savo, Graph-based ontology classification in OWL 2 QL, in: *Proceeding of 10th International Conference on the Semantic Web: Semantics and Big Data (ESWC)*, 2013, pp. 320–334.
- [17] M. Akram, Bipolar fuzzy graphs with applications, *Knowl. Based Syst.* 39 (2013) 1–8.
- [18] N. Nikitina, S. Rudolph, B. Glimm, Interactive ontology revision, *J. Web Sem.* 12 (1) (2012) 118–130.
- [19] M. Lenzerini, D.F. Savo, Updating inconsistent description logic knowledge bases, in: *Proceeding of 20th European Conference on Artificial Intelligence (ECAI)*, 2012, pp. 516–521.
- [20] F. Baader, R. Peñaloza, B. Suntisrivaraporn, Pinpointing in the description logic  $\mathcal{EL}^+$ , in: *Proceeding of Advances in Artificial Intelligence (KI)*, 2007, pp. 52–67.
- [21] A. Kalyanpur, B. Parsia, M. Horridge, E. Sirin, Finding all justifications of OWL DL entailments, in: *Proceedings of Sixth International Semantic Web Conference (ISWC) and Second Asian Semantic Web Conference (ASWC)*, 2007, pp. 267–280.
- [22] G. Qi, A. Hunter, Measuring incoherence in description logic-based ontologies, in: *Proceedings of Sixth International Semantic Web Conference (ISWC) and Second Asian Semantic Web Conference (ASWC)*, 2007, pp. 381–394.
- [23] R. Reiter, A theory of diagnosis from first principles, *Artif. Intell.* 32 (1) (1987) 57–95.
- [24] S. Schlobach, Diagnosing terminologies, in: *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, 2005, pp. 670–675.
- [25] M.A. Casteleiro, J.J.D. Diz, Clinical practice guidelines: a case study of combining OWL-S, OWL, and SWRL, *Knowl. Based Syst.* 21 (3) (2008) 247–255.
- [26] I. Robinson, J. Webber, E. Eifrem, Graph Databases, O'Reilly Media, Inc., pp. 25–63.
- [27] J. Webber, A programmatic introduction to Neo4j, in: *Proceedings of Conference on Systems, Programming, and Applications: Software for Humanity (SPLASH)*, 2012, pp. 217–218.
- [28] F. Baader, R. Peñaloza, Automata-based axiom pinpointing, in: *Proceeding of Fourth International Joint Conference on Automated Reasoning (IJCAR)*, 2008, pp. 226–241.
- [29] L. Zhou, H. Huang, G. Qi, Y. Qu, Q. Ji, An algorithm for calculating minimal unsatisfiability-preserving subsets of ontology in DL-Lite, *J. Comput. Res. Dev.* 48 (3) (2011) 2334–2342.
- [30] G. Friedrich, K.M. Shchekotykhin, A general diagnosis method for ontologies, in: *Proceeding of Fourth International Semantic Web Conference (ISWC)*, 2005, pp. 232–246.
- [31] U. Junker, QuickXplain: preferred explanations and relaxations for over-constrained problems, in: *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, 2004, pp. 167–172.
- [32] Q. Ji, Z. Gao, Z. Huang, M. Zhu, Measuring effectiveness of ontology debugging systems, *Knowl. Based Syst.* 71 (2014) 169–186.
- [33] T.A. Meyer, K. Lee, R. Booth, J.Z. Pan, Finding maximally satisfiable terminologies for the description logic  $\mathcal{ALC}$ , in: *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006, pp. 269–274.
- [34] J. Du, G. Qi, X. Fu, A practical fine-grained approach to resolving incoherent OWL 2 DL terminologies, in: *Proceedings of the 23rd International Conference on Information and Knowledge Management (CIKM)*, 2014, pp. 919–928.
- [35] G. Qi, J. Du, Model-based revision operators for terminologies in description logics, in: *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009, pp. 891–897.
- [36] B.C. Grau, E.J. Ruiz, E. Kharlamov, D. Zhelenyakov, Ontology evolution under semantic constraints, in: *Proceeding of 13th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2012, pp. 137–147.

- [37] Z. Wang, K. Wang, R.W. Topor, A new approach to knowledge base revision in DL-Lite, in: Proceeding of the 24th National Conference on Artificial Intelligence (AAAI), 2010, pp. 369–374.
- [38] Z. Zhuang, Z. Wang, K. Wang, G. Qi, Contraction and revision over DL-Lite TBoxes, in: Proceedings of the 28th National Conference on Artificial Intelligence (AAAI), 2014, pp. 1149–1156.
- [39] Z. Wang, K. Wang, Z. Zhuang, G. Qi, Instance-driven ontology evolution in DL-Lite, in: Proceedings of the 29th National Conference on Artificial Intelligence (AAAI), 2015, pp. 1656–1662.
- [40] G. Qi, Z. Wang, K. Wang, X. Fu, Z. Zhuang, Approximating model-based ABox revision in DL-Lite: Theory and practice, in: Proceedings of the 29th National Conference on Artificial Intelligence (AAAI), 2015, pp. 254–260.