Paraconsistent Query Answering Over DL-Lite Ontologies

Liping Zhou a,∗, Houkuan Huang a, Guilin Qi b, Yue Ma c, Zhisheng Huang d, and Youli Qu a
a School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
E-mail:{06112065, hkhuang, ylqu}@bju.edu.cn
b School of Computer Science and Engineering, Southeast University, Nanjing, China
E-mail: gqi@seu.edu.cn
c Laboratoire d’Informatique de l’université Paris-Nord (LIPN) - UMR 7030, Université Paris 13 - CNRS, France
E-mail: Yue.Ma@lipn.univ-paris13.fr
d Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
E-mail: huang@cs.vu.nl

Abstract. Consistent query answering over description logic-based ontologies is an important topic in ontology engineering as it can provide meaningful answers to queries posed over inconsistent ontologies. Current approaches for dealing with this problem usually consist of two steps: the first is to extract some consistent sub-ontologies of an inconsistent ontology, and then the second is to pose the query over these consistent sub-ontologies. In this paper, we propose an alternative approach to consistent query answering for conjunctive queries over DL-Lite ontologies based on four-valued semantics, where DL-Lite is a family of tractable Description Logics (DLs). We give an algorithm to compute answers to a query over inconsistent DL-Lite ontologies and show that it is tractable. In particular, it is proven to be in PTime with respect to the size of the TBox, and LOGSPACE with respect to the size of the ABox.

Keywords: Description Logics, DL-Lite, Inconsistency handling, Paraconsistency, query answering

1. Introduction

As a shared conceptualization of a particular domain, ontology plays an important role for the success of the Semantic Web [3]. Description Logics (DLs) are considered as important formal description languages for specifying ontologies. They provide logical underpinning of Web Ontology Language (OWL). For example, the expressive DL $\text{SHOIN}(D)$ underpins OWL DL, which is the key language of OWL. However, expressive DLs suffer from worst-case exponential time behavior of reasoning [2] which may hinder practical applications to very large real life ontologies. As an important tractable DL family, DL-Lite can keep standard reasoning tasks within polynomial time complexity in the size of the ontology [6].

However, as the sizes of ontologies grow and applications become more complex, inconsistencies frequently occur within the ontology lifecycle, such as ontology construction, ontology evolution and ontology merging. Conclusions drawn from an inconsistent ontology by classical inference may be completely meaningless according to the fact $\text{ex contradictione quodlibet}$ [8]. Generally, there are two main ways to deal with inconsistencies in ontologies. One is to diagnose and repair it when we encounter an inconsistency [16]. The other approach is to simply avoid the inconsistency and to apply a non-standard reasoning method to obtain meaningful answers [8,12].

Following the second approach, consistent query answering over inconsistent databases or inconsistent propositional knowledge bases has been seen as an
important research area. Many methods are known [17,1,5,10]. Arenas el al. have provided a method for consistent query answers in inconsistent databases by repairing a database instance that may violate integrity constraints specified over its schema [1]. Calì et al. in [5] have shown that consistent query answering of conjunctive queries expressed over database schemas with (even simple forms of) integrity constraints is a coNP-complete problem in data complexity [10]. Lembo et al. [10] have proposed an approach for consistent instance checking based on an inconsistency-tolerant semantics. Their method relies on the notion of repair that deletes some inconsistent membership assertions, which will lead to information loss. Lembo et al. [10] have shown that their approach to consistent query answering for conjunctive queries over inconsistent DL-Lite ontologies is general intractable w.r.t. data complexity. Since DL-Lite is a tractable DLs which is suitable for dealing with large amount of data, finding a tractable approach to consistent query answering for conjunctive queries over inconsistent DL-Lite ontologies is necessary, which is studied in this paper.

In this paper, we propose a novel approach to tractable consistent query answering for conjunctive queries over DL-Lite ontologies which is based on four-valued semantics [15,12]. To this end, we first present a four-valued semantics for DL-Lite. And then, by extending the notion of chase in [6], we give a notion of four-valued semantics 4-chase which is used to construct a four-valued model for DL-Lite ontology. Then, we present an algorithm for consistent query answering for conjunctive queries over DL-Lite and show that its computational complexity is LOGSPACE with respect to the size of the ABox. Finally, the advantages of our approach is shown by a theoretical and experimental comparison between our approach and an alternative approach to consistent query answering based on four-valued semantics.

The rest of the paper is organized as follows: Section 2 presents some basic notions for DL-Lite. Section 3 gives a four-valued semantics for DL-Lite. Section 4 presents an algorithm to compute a four-valued model and to compute the certain answers to a query over an inconsistent DL-Lite ontology. Comparison with an alternative approach to consistent query answering based on four-valued semantics is reported in Section 5. We give related work in Section 6 and conclude our paper in Section 7.

2. Preliminaries

DL-Lite is a family of Description Logics (DLs) used to capture some of the most popular conceptual modeling formalisms, such as Entity-Relationship model and UML class diagrams, while preserving the tractability of the most important reasoning tasks, such as ontology satisfiability. We mainly consider the following important DLs in DL-Lite family: DL-Lite, and DL-LiteR [6].

At the core of both DL-Lite and DL-LiteR, concepts and roles are formed according to the following syntax:

\[ B \rightarrow A \mid \exists R \mid R \rightarrow P \mid P^- \]
\[ C \rightarrow B \mid \neg B \mid E \rightarrow R \mid \neg R \]

where \( A \) and \( P \) denote an atomic concept and an atomic role respectively; \( B \) denotes a basic concept (i.e., a concept of the form \( A, \exists R \)); \( R \) denotes a basic role (i.e., a role of the form \( P, P^- \)), where \( P^- \) denotes the inverse of the atomic role \( P \); \( C \) denotes a general concept (i.e., a concept of the form \( B, \neg B \)), whereas \( E \) denotes a general role (i.e., a concept of the form \( R, \neg R \)).

A DL-Lite or DL-LiteR ontology consists of a TBox and an ABox. A DL-LiteR TBox is formed by a set of concept inclusion axioms of the form \( B \sqsubseteq C \) and role inclusion axioms of the form \( R \sqsubseteq E \). DL-LiteTBox is formed by a set of concept inclusion axioms and functionality assertions of the form (func \( P \)) or (func \( P^- \)). DL-LiteF and DL-LiteR ABoxes are formed by a finite set of membership assertions on atomic concepts and atomic roles, of the form \( A(a), P(a, b) \), where \( a \) and \( b \) are constants. Hereinafter, we use the term DL-Lite to refer to either DL-LiteR or DL-LiteF, we call positive inclusions(PIs) assertions of the form \( B_1 \sqsubseteq B_2 \) or of the form \( R_1 \sqsubseteq R_2 \), whereas we call negative inclusions(NIs) assertions of the form \( B_1 \not\sqsubseteq B_2 \) or \( R_1 \not\sqsubseteq R_2 \).

The semantics of DL-Lite is defined via a model-theoretic semantics, which explicates the relationship between the language syntax and the model of a domain: An interpretation \( I = (\Delta^I, \cdot^I) \), consisting of a non-empty interpretation domain \( \Delta^I \) and an interpretation function \( \cdot^I \), which maps from individuals, concepts and roles to elements of the domain, subsets of the domain and binary relations on the domain, respectively. Tabel 1 gives the semantics of DL-Lite.

An interpretation \( I \) is called a model of an ontology \( O \), iff it satisfies each axiom and each assertion in \( O \). An ontology is satisfiable if it has at least one model. An ontology \( K \) logically implies an assertion
which is defined as follows:

\[ \Delta^i = \Delta \setminus \{ \alpha \} \]

The unique name assumption on constants [2] is adapted by DL-Lite. Furthermore, DL-Lite_\mathcal{F} has the finite model property, that is, if a DL-Lite_\mathcal{F} is consistent, then it has a classical model whose domain is finite [2,6]. However DL-Lite_\mathcal{F} does not have finite model property [6].

Calvanese et al. [6] have given a specific interpretation about ABox \( \mathcal{A} \) denoted as \( db(\mathcal{A}) = (\Delta db(\mathcal{A}), \omega db(\mathcal{A})) \) which is defined as follows:

- \( \Delta db(\mathcal{A}) \) is the nonempty set consisting of all constants occurring in \( \mathcal{A} \);
- \( \omega db(\mathcal{A}) = a \), for each constant \( a \);
- \( A db(\mathcal{A}) = \{ a \mid A(a) \in \mathcal{A} \} \), for each atomic concept \( A \);
- \( P db(\mathcal{A}) = \{ (a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A} \} \), for each atomic role \( P \).

A union of conjunctive queries (UCQ) \( q \) over a DL-Lite ontology \( \mathcal{K} \) is an expression of the form \( q(\vec{x}) \rightarrow \bigvee_{i=1}^{n} \exists y_i . \text{conj}_i(\vec{x}, y_i) \), where each \( \text{conj}_i(\vec{x}, y_i) \) is a conjunction of atoms and equalities and \( \vec{x} \) and \( \vec{y} \) are vectors of distinguished variables and non-distinguished variables respectively [6,10]. The size of \( \vec{x} \) is called the arity of \( q \). Atoms in each \( \text{conj}_i \) are of the form \( A(z) \) or \( P(z_1, z_2) \), where \( A \) and \( P \) are respectively atomic concepts and atomic roles of \( \mathcal{K} \), \( z, z_1 \) and \( z_2 \) are either constants in \( \mathcal{K} \) or variables. A Boolean UCQ is a query with arity 0, written simply as a sentence of the form \( \bigvee_{i=1}^{n} \exists y_i . \text{conj}_i(\vec{y}) \). A UCQ with a single conjunction of atoms is called conjunctive query (CQ). Let \( q \) be a Boolean UCQ over a DL-Lite ontology \( \mathcal{K} \). We say that \( q \) is entailed by \( \mathcal{K} \), and write \( \mathcal{K} \models q \), if, for every model \( \mathcal{M} \) of \( \mathcal{K} \), \( \mathcal{M} \models q \).

### Table 1

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{A} )</td>
<td>( \Delta^i \subseteq \Delta^j )</td>
</tr>
<tr>
<td>( \exists R )</td>
<td>( { d \mid \exists (d, e) \in R^i } )</td>
</tr>
<tr>
<td>( \neg A )</td>
<td>( \Delta^i \setminus { \alpha } )</td>
</tr>
<tr>
<td>( \neg \exists R )</td>
<td>( \Delta^j \setminus (\exists R)^i )</td>
</tr>
<tr>
<td>( P )</td>
<td>( P^j \subseteq \Delta^i \times \Delta^j )</td>
</tr>
<tr>
<td>( P^- )</td>
<td>( { (o, o') \mid (o, o') \in P^j } )</td>
</tr>
<tr>
<td>( \neg R )</td>
<td>( (\Delta^i \times \Delta^j) \setminus R^j )</td>
</tr>
<tr>
<td>( B \subseteq C )</td>
<td>( B^j \subseteq C^j )</td>
</tr>
<tr>
<td>( R \subseteq S )</td>
<td>( R^j \subseteq S^j )</td>
</tr>
<tr>
<td>(func ( R ))</td>
<td>( \forall d, e, e', (d, e) \in R^j \land (d, e') \in R^j \implies e = e' )</td>
</tr>
<tr>
<td>( A(a) )</td>
<td>( a^j \in \Delta^j )</td>
</tr>
<tr>
<td>( P(a, b) )</td>
<td>( (a^j, b^j) \in P^j )</td>
</tr>
</tbody>
</table>

Consider a query \( q(x) \rightarrow \text{Stud}(x) \). Now, let us see the execution of the algorithm PerfectRef. Since the atom \( \text{Stud}(x) \) can be applied to the PI \( \text{PhDStud} \subseteq \text{Stud} \), we obtain a new query \( q(x) \rightarrow \text{PhDStud}(x) \). So the result returned by the algorithm is the union of the following: \( q(x) \rightarrow \text{Stud}(x), q(x) \rightarrow \text{PhDStud}(x) \).

After query rewriting by DL-Lite reasoners [6], query answering over DL-Lite ontologies can be carried out by an SQL engine, so as to take advantage of existing query optimization strategies and algorithms provided by modern database management systems.

3. Four-valued semantics for DL-Lite

In this section, we present the four-valued semantics for DL-Lite given in [11,18], which will be used to define the problem of paraconsistent query answering over DL-Lite ontologies.

For a given domain \( \Delta \) and a concept \( A \) (resp., \( R \), a four-valued interpretation over \( \Delta \) assigns to \( A \)
(resp., \( R \)) an extended truth value \( \langle A_P, A_N \rangle \) (resp., \( \langle R_P, R_N \rangle \)) where \( A_P \) is the subset of \( \Delta \) (resp., \( R_P \)) to be true and \( A_N \) is the subset of \( \Delta \) (resp., \( R_N \)) that supports \( A \) (resp., \( R \)) to be false. We denote \( \text{proj}^+(\langle P, N \rangle) = P \) and \( \text{proj}^-(\langle P, N \rangle) = N \) [12]. The four-valued semantics of DL-Lite is given by means of an interpretation \( I = \langle \Delta_I, I \rangle \) consisting of a non-empty interpretation domain \( \Delta_I \) and an interpretation function \( I \) satisfying the conditions [13] in Table 2. In Table 2, we assign to "=\(=\)" a para-equality w.r.t. the four-valued semantics. That is, for any given domain \( \Delta \), we assign to "\(=\)", "\(\neq\)", an extended truth value \( \langle =_P, =_N \rangle \), where \( =_P \) stands for the set of pairs of constants which are equal and \( =_N \) stands for the set of pairs of constants which are not equal. Based on the semantics of "\(=\)" and "\(\neq\)", we know it has some properties of classical equality, such as reflexive, symmetric and transitive; For "\(=\)", we know it has symmetric property, that is, if \( (a, b) \in =_N \), then \( (b, a) \in =_N \). For simplicity, we do not consider these properties of "\(=\)" and "\(\neq\)" in this paper. The UNA can be expressed as \( \forall x, y \in \Delta^\text{th}(A), (x, y) \in \text{proj}^-\langle =_I \rangle \).

Based on the four-valued semantics, there are four truth values for membership assertions. The four truth values are true, false, contradictory and unknown, we use the symbols \( t, f, \mathbb{B}, \mathbb{N} \) to denote them respectively [12]. The truth value \( \mathbb{B} \) can be understood to stand for true and false and it can be used to deal with inconsistent knowledge, while \( \mathbb{N} \) stands for neither true nor false, i.e. for the absence of any information about truth or falsity. The corresponding four-valued semantics for concept assertions is given as follows:

\[ A^I(a) = t, \text{ iff } a \in \text{proj}^+(A^I) \text{ and } a \notin \text{proj}^-(A^I); \]
\[ A^I(a) = f, \text{ iff } a \notin \text{proj}^+(A^I) \text{ and } a \in \text{proj}^-(A^I); \]
\[ A^I(a) = \mathbb{B}, \text{ iff } a \notin \text{proj}^+(A^I) \text{ and } a \in \text{proj}^-(A^I); \]
\[ A^I(a) = \mathbb{N}, \text{ iff } a \notin \text{proj}^+(A^I) \text{ and } a \notin \text{proj}^-(A^I). \]

The corresponding four-valued semantics for role assertion (or equality "\(=\)") can be defined in a similar way.

Given a four-valued interpretation \( I \), we say that \( I \) satisfies an axiom:

\[ \begin{align*}
&- B \subseteq C \text{ if } \text{proj}^+(B^I) \subseteq \text{proj}^+(C^I); \\
&- R_1 \subseteq R_2 \text{ if } \text{proj}^+(R_1^I) \subseteq \text{proj}^+(R_2^I); \\
&- (\text{funct } P) \text{ if } \forall x, y, z, (x, y) \in \text{proj}^+(P^I) \land (x, z) \in \text{proj}^+(P^I) \rightarrow (y, z) \in \text{proj}^+(\mathbb{N}); \\
&- A(a) \text{ if } a^I \in \text{proj}^+(A^I); \\
&- P(a, b) \text{ if } (a^I, b^I) \in \text{proj}^+(P^I). 
\end{align*} \]

A four-valued model of a DL-Lite ontology \( K \) is a four-valued interpretation \( I \) which satisfies each assertion and each axiom in \( K \). A DL-Lite ontology is four-valued satisfiable (unsatisfiable) if there exists (does not exist) such a model. A DL-Lite ontology \( K \) logically implies an assertion \( \alpha \) under four-valued semantics, written \( K \models \alpha \), if all four-valued models of \( K \) are also four-valued models of \( \alpha \).

**Example 2** [18] Given a DL-Lite ontology \( K = \langle T, A \rangle \), where:

\[
T = \{ \text{PhDStud} \sqsubseteq \text{Stud}, \text{PhDStud} \sqsubseteq \text{Employee}, \text{Stud} \sqsubseteq \neg \text{Employee}, \text{Stud} \sqsubseteq \exists \text{hasTutor}, \text{(funct hasTutor)} \},
\]

\[
A = \{ \text{PhDStud}(a), \text{hasTutor}(a, b), \text{hasTutor}(a, c) \}. \]

We can find that it is an inconsistent ontology. Consider the following four-valued interpretation \( I = \langle \Delta^I, I \rangle \), where:

\[
\Delta^I = \{ a, b, c \}, \quad \text{hasTutor}^I = \{ \{ a, c \}, \{ a, b \} \}, \quad \text{Stud}^I = \{ \{ a \} \}, \quad \text{PhDStud}^I = \{ \{ a \} \}, \quad \text{Employee}^I = \{ \{ a \} \},
\]

\[
\{ (a, b), (a, c), (b, c) \}, \quad \{ (a, b), (a, c), (b, c) \}.
\]

We can find that \( I \) is a four-valued model of \( K \) and \( \text{PhDStud}^I(a) = t, \text{Stud}^I(a) = t, \text{Employee}^I(a) = \mathbb{B} \).
and \(=^I(b, c) = B\). It is easy to obtain four-valued semantics for other atomic assertions.

Under the four-valued semantics, we will use a special membership assertion of the form \(\neg A(a)\) or \(\neg R(a, b)\), which is not supported in DL-Lite under the classical semantics. In the following, we call an assertion of the form \(A(a)\) or of the form \(P(a, b)\) a positive membership assertion (PMA), whereas we call an assertion of the form \(\neg A(a), \neg R(a, b)\) a negative membership assertion (NMA). For simplicity, we use the term “membership assertions” to refer to PMAs or NMAs. We will use the symbol “\(\ast\)” to denote all constants in the domain \(A\). For example, assume a domain \(A = \{a, b, c\}\), then \(R(a, \ast)\) denotes the set \(\{R(a, a), R(a, b), R(a, c)\}\). For easy illustration, we also use the function \(ga\) [6] that takes as input a basic role and two constants and returns a membership as-

\[R \models \{m\}\]

tance for other atomic assertions.

We will use the symbol “\(\ast\)” to refer to PMAs or NMAs. We will use the symbol “\(\ast\)” to denote all constants in the domain \(A\). For example, assume a domain \(A = \{a, b, c\}\), then \(R(a, \ast)\) denotes the set \(\{R(a, a), R(a, b), R(a, c)\}\). For easy illustration, we also use the function \(ga\) [6] that takes as input a basic role and two constants and returns a membership assertion, that is, if \(R = P\), then \(ga(R, a, b) = P(a, b)\); if \(R = P^\ast\), then \(ga(R, a, b) = P(b, a)\).

4. Query answering over DL-Lite based on four-valued semantics

In this section, we mainly discuss the problem of paraconsistent query answering over an inconsistent DL-Lite ontology. We propose a tractable method to query answering over inconsistent DL-Lite ontologies based on four-valued semantics.

4.1. Four-valued canonical interpretation

Let us first define the answers and the certain an-

\[\alpha = A_1 \sqsubseteq A_2, f = A_1(a) \land A_2(a) \notin S;\]
\[\alpha = A \sqsubseteq \exists R, f = A(a) \land \text{there does not exist any constant } b \text{ such that } ga(R, a, b) \in S;\]
\[\alpha = \exists R_1 \sqsubseteq \exists R_2, f = ga(R_1, a, b) \land \text{there does not exist any constant } c \text{ such that } ga(R_2, a, c) \in S;\]
\[\alpha = R_1 \sqsubseteq R_2, f = R_1(a, b) \land ga(R_2, a, b) \notin S;\]
\[\alpha = A \sqsubseteq \neg \exists R, f = A(a) \land \text{there exists a constant } b \text{ such that } ga(R, a, b) \in S;\]
\[\alpha = \exists R_1 \sqsubseteq \neg \exists R_2, f = ga(R_1, a, b) \land \text{there exists a constant } c \text{ such that } \neg ga(R_2, a, c) \in S;\]
\[\alpha = \text{funct}(R), f = ga(R, a, b) \land \text{there exists a constant } x (x \neq b) \text{ such that } ga(R, a, x) \in S, = (b, x) \notin S;\]

Notice that by Definition 2, \(4\text{-Ans}(q, K)\) is finite because \(K\) is finite which makes the number of constants appearing in \(A\) finite. If \(q\) is a Boolean query over a DL-Lite \(K\), we only need to judge whether \(K \models q\) under four-valued semantics.

Now, we introduce the construction of four-valued canonical interpretation which will be used in the sequel. First, we need to extend some definitions in [6]. We start by defining the notion of applicable inclusion or applicable functionality. Then we give a definition of \(4\text{-chase}(K)\) by extending the notion of chase [6] which is defined based on classical semantics. The four-valued canonical interpretation of a DL-Lite ontol-

ogy is a four-valued interpretation constructed ac-

\[\Delta \ast \text{constants in the domain } \ast \text{or NMAs. We will use the symbol ”}\ast\text{“ to refer to PMAs}\]

\[4\text{-Ans}(K, \langle T, A \rangle) \text{ be a DL-Lite ontology. Suppose } S \text{ is a set of membership assertions. An inclusion assertion or a functionality assertion } \alpha \in T \text{ is applicable in } S \text{ to a membership assertion } f \in S \text{ if}\]

\[\alpha = A_1 \sqsubseteq A_2, f = A_1(a) \land A_2(a) \notin S;\]
\[\alpha = A \sqsubseteq \exists R, f = A(a) \land \text{there does not exist any constant } b \text{ such that } ga(R, a, b) \in S;\]
\[\alpha = \exists R_1 \sqsubseteq \exists R_2, f = ga(R_1, a, b) \land \text{there does not exist any constant } c \text{ such that } ga(R_2, a, c) \in S;\]
\[\alpha = R_1 \sqsubseteq R_2, f = R_1(a, b) \land ga(R_2, a, b) \notin S;\]
\[\alpha = A \sqsubseteq \neg \exists R, f = A(a) \land \text{there exists a constant } b \text{ such that } ga(R, a, b) \in S;\]
\[\alpha = \exists R_1 \sqsubseteq \neg \exists R_2, f = ga(R_1, a, b) \land \text{there exists a constant } c \text{ such that } \neg ga(R_2, a, c) \in S;\]
\[\alpha = \text{funct}(R), f = ga(R, a, b) \land \text{there exists a constant } x (x \neq b) \text{ such that } ga(R, a, x) \in S, = (b, x) \notin S;\]

In fact, Definition 3 extends Definition 4 of [6] in which only PIs in TBox are involved and Definition 8 of [18] in which only NIs and functionality assertions logically implied by TBox are involved. In contrast, Definition 3 involves all axioms in TBox.

We will use Applicable inclusion or functionality assertion to construct \(4\text{-chase}(K)\). \(4\text{-chase}(K)\) is equal to \(\mathcal{A}\) at the initial state. Then at each step of construction, an inclusion assertion or a functionality assertion \(\alpha \in T\) is applied to a membership assertion
$f \in S$, which means adding a new suitable membership assertion to $S$, thus obtaining a new set $S'$ in which $\alpha$ is no longer applicable to $f$. For example, if $\alpha = A_1 \subseteq \neg A_2$ is applicable to $f = A_1(a)$, then the membership assertion to be added to $S$ is $\neg A_2(a)$, that is, $S' = S \cup \neg A_2(a)$.

Since the first five rules of Definition 3 is the same as Definition 4 in [6], based on [6], we know that the construction process of $4$-chase($K$) also depends on the order in which we select the inclusion axiom, the membership assertion and constants we introduce at each step. Like Calvanese et al. have discussed in [6], we denote with $\Gamma_A$ the set of all constant symbols occurring in $A$ and we assume that we have an infinite set $\Gamma_n$ of constant symbols not occurring in $A$, such that the set $\Gamma_C = \Gamma_A \cup \Gamma_n$ is totally ordered in lexicographic way. And we select all axioms in $T$, membership assertions, constants symbols in lexicographic order. Then our notion of $4$-chase($K$) is precisely given below.

**Definition 4** Let $K = \langle T, A \rangle$ be a DL-Lite ontology, $n$ be the number of membership assertions in $A$, and $\Gamma_n$ be the set of constants defined above. Assume that the membership assertions in $A$ are numbered from 1 to $n$ following their lexicographic order, we have the definition as follows: $S_0 = A$ and $S_{j + 1} = S_j \cup f_{new}$, where $f_{new}$ is a membership assertion numbered with $n + j + 1$ in $S_{j + 1}$ and obtained as follows: Let $f$ be the first membership assertion in $S_j$ such that there exists an axiom $\alpha \in T$ applicable in $S_j$ to $f$. Let $\alpha$ be the lexicographically first inclusion or functionality assertion applicable in $S_j$ to $f$. Let $a_{new}$ be the constant of $\Gamma_n$ that follows lexicographically all constants occurring in $S_j$. Case $\alpha$, $f$ of

- (cr1) $\alpha = A_1 \not\subseteq A_2$ and $f = A_1(a)$ then $f_{new} = A_2(a)$
- (cr2) $\alpha = A \not\subseteq \exists R$ and $f = A(a)$ then $f_{new} = ga(R, a, a_{new})$
- (cr3) $\alpha = \exists R \not\subseteq A$ and $f = ga(R_1, a, b)$ then $f_{new} = A(a)$
- (cr4) $\alpha = \exists R_1 \not\subseteq \exists R_2$ and $f = ga(R_1, a, b)$ then $f_{new} = ga(R_2, a, a_{new})$
- (cr5) $\alpha = R_1 \not\subseteq R_2$ and $f = ga(R_1, a, b)$ then $f_{new} = ga(R_2, a, b)$
- (cr6) $\alpha = A_1 \not\subseteq \neg A_2$ and $f = A_1(a)$ then $f_{new} = \neg A_2(a)$
- (cr7) $\alpha = \exists R \not\subseteq \neg A$ and $f = ga(R, a, b)$ then $f_{new} = \neg A(a)$
- (cr8) $\alpha = R_1 \not\subseteq \neg R_2$ and $f = ga(R_1, a, b)$ then $f_{new} = \neg ga(R_2, a, b)

- (cr9) $\alpha = A \not\subseteq \exists R$ and $f = A(a)$ then $f_{new} = \neg ga(R, a, *)$
- (cr10) $\alpha = \exists R_1 \not\subseteq \exists R_2$ and $f = ga(R_1, a, b)$ then $f_{new} = \neg ga(R_2, a, *)$
- (cr11) $\alpha = (\text{funct } R)$ and $f = ga(R, a, b), \forall x, ga(R, a, x) \in S_j$ and $b \neq x$ then $f_{new} = \ast(b, x)$.

Then $4$-chase($K$) is the set of membership assertions obtained as the infinite union of all $S_j$, that is, $4$-chase($K$) = $\bigcup_{j \in \mathbb{N}} S_j$.

Note that not only PIs but also NIs and functionality assertions in $K$ have a role in constructing $4$-chase($K$). The difference between chase in [6] and $4$-chase is illustrated in Figure 1.

![Fig. 1. the difference between chase in [6] and 4-chase](image)

Furthermore, each inclusion or functionality assertion in $T$ can be applied at most once to a membership assertion (afterwards, the precondition is not satisfied and the inclusion or functionality assertion is no longer applicable). In the following, we will denote with $4$-chase$_{i}(K)$ the portion of the chase obtained after $i$ applications of the chase rules, selected according to the ordering established in Definition 4, that is, $4$-chase$_{i}(K) = \bigcup_{j \in \{0, \ldots, i\}} S_j$. We can easily obtain the following proposition which is similar to Proposition 6 in [6].

**Proposition 1** Let $K = \langle T, A \rangle$ be a DL-Lite ontology, and let $\alpha$ be an inclusion assertion or a functionality assertion in $T$. Then, if there is an $i \in \mathbb{N}$ such that $\alpha$ is applicable in $4$-chase$_{i}(K)$ to a membership assertion $f \in 4$-chase$_{i}(K)$, then there is a $j \geq i$ such that $4$-chase$_{j+1}(K) = 4$-chase$_{j}(K) \cup f'$, where $f'$ is the result of applying $\alpha$ to $f$ in $4$-chase$_{j}(K)$.

**Proof.** This proof is analogous to the proof of Proposition 6 in [6].

Now, with the notion of $4$-chase($K$), we define the four-valued canonical interpretation, denoted as $4$-can($K$) = $\langle \Delta^{4\text{-can}}(K), \ast^{4\text{-can}}(K) \rangle$, where

- $\Delta^{4\text{-can}}(K) = \Gamma_C$,
- $\ast^{4\text{-can}}(K) = a$, for each constant $a$ occurring in $4$-chase($K$).
Proof. Considering all possible cases.

Like \( \text{can} \) in [6], we also define \( 4\text{-can}(K) = (\Delta^4\text{-can}(K), \text{can}(K)) \), where \( 4\text{-can}(K) \) is analogous to \( 4\text{-can}(K) \) but it refers to \( 4\text{-chase}(K) \) instead of \( 4\text{-chase}(K) \). \( 4\text{-chase}(K) \) and \( 4\text{-can}(K) \) (resp., \( 4\text{-can}(K) \) and \( 4\text{-can}(K) \)) are strongly connected. It is easy to see that \( 4\text{-can}(K) \) (resp., \( 4\text{-can}(K) \)) is also unique.

Now we give a four-valued interpretation \( 4\text{-db}(A) = (\Delta^4\text{-db}(A), \Delta^4\text{-db}(A)) \), where

- \( \Delta^4\text{-db}(A) = \Delta\text{-db}(A) \),
- \( a^{4\text{-db}(A)} = a \), for each constant \( a \) occurring in \( A \),
- \( A^{4\text{-db}(A)} = \{ \{ \{ a \mid A(a) \in A \} \}, \{ \emptyset \} \} \), for each atomic concept \( A \),
- \( P^{4\text{-db}(A)} = \{ \{ \{ a, a \} \mid P(a, a) \in A \} \}, \{ \emptyset \} \}\), for each atomic role \( P \),
- \( \neg^{4\text{-db}(A)} = \{ \{ \{ a, a \} \mid \forall a \in \Delta^4\text{-db}(A) \}, \{ \{ b, b \} \mid \forall b, b \in \Delta^4\text{-db}(A) \} \} \).

Note that \( 4\text{-can}(K) \) is tightly related to the interpretation \( 4\text{-db}(A) \), that is, \( 4\text{-db}(A) = 4\text{-can}(K) \).

The following theorem shows a notable property of \( 4\text{-can}(K) \).

**Theorem 2** Let \( K = (T, A) \) be a DL-Lite ontology. Then \( 4\text{-can}(K) \) is a four-valued model of \( (T, A) \).

**Proof.** This proof is analogous to the proof of Lemma 7 in [6]. First \( 4\text{-can}(K) \) satisfies all membership assertions in \( A \) because \( A \subseteq 4\text{-chase}(K) \). So we only need to prove that \( 4\text{-can}(K) \models T \). Let us proceed by contradiction considering all possible cases.

For the inclusion assertions such that \( A_1 \subseteq A_2, \exists R \subseteq A, A \subseteq \exists R, \exists R \subseteq R_1 \subseteq R_2 \) can be proved in an analogous way of the proof of Lemma 7 in [6].

Now we assume that an inclusion assertion of the form \( A_1 \subseteq \neg A_2 \in T \), where \( A_1 \) and \( A_2 \) are atomic concepts, is not satisfied by \( 4\text{-can}(K) \). Then there exists a constant \( a \in \Gamma_C \) such that \( A_1(a) \in 4\text{-chase}(K) \) and \( \neg A_2(a) \notin 4\text{-chase}(K) \). However, such a situation would trigger the rule \( \text{cr6} \), \( \neg A_2(a) \) will be added to \( 4\text{-chase}(K) \) at some step, hence contradicting the assumption. For the inclusion assertions such that \( \exists R \subseteq \neg A \) can be proved in an analogous way.

Now we assume by contradiction that an inclusion assertion of the form \( \exists R_1 \subseteq \neg \exists R_2 \in T \), where \( R_1 \) and \( R_2 \) are atomic role, is not satisfied by \( 4\text{-can}(K) \). Then there exists a pair of constants \( a, b \in \Gamma_C \) such that \( ga(R_1, a, b) \in 4\text{-chase}(K) \) and there exist a constant \( c \in \Gamma_C \) such that \( \neg ga(R_2, a, c) \notin 4\text{-chase}(K) \). However, such a situation would trigger the rule \( \text{cr10} \), \( \neg ga(R_2, a, c) \) will be added to \( 4\text{-chase}(K) \) at some step, hence contradicting the assumption. For the inclusion assertions such that \( A \subseteq \neg \exists R \) can be proved in an analogous way.

Now we assume by contradiction that an inclusion assertion of the form \( R_1 \subseteq \neg R_2 \in T \), where \( R_1 \) and \( R_2 \) are atomic role, is not satisfied by \( 4\text{-can}(K) \). Then there exists a pair of constants \( a, b \in \Gamma_C \) such that \( ga(R_1, a, b) \in 4\text{-chase}(K) \) and \( \neg ga(R_2, a, b) \notin 4\text{-chase}(K) \). However, such a situation would trigger the rule \( \text{cr8} \), \( \neg ga(R_2, a, b) \) will be added to \( 4\text{-chase}(K) \) at some step, hence contradicting the assumption. Finally, assume by contradiction that a functionality assertion of the form \( \text{fun}(R) \), where \( R \) is a basic role, is not satisfied by \( 4\text{-can}(K) \). Then there exists constants \( a, b, c \in \Gamma_C \) such that \( ga(R_1, a, b) \in 4\text{-chase}(K) \) and \( \neg ga(R_2, a, b) \notin 4\text{-chase}(K) \). However, such a situation would trigger the rule \( \text{cr10} \), \( \neg ga(R_2, a, b) \) will be added to \( 4\text{-chase}(K) \) at some step, hence contradicting the assumption.

Theorem 2 tells us that for any DL-Lite ontology, we can always construct a four-valued model for it, namely \( 4\text{-can}(K) \). In the following, we will use \( 4\text{-can}(K) \) for paraconsistent query answering over DL-Lite ontologies.

4.2. Paraconsistent query answering

In this section, we consider paraconsistent query answering over DL-Lite ontologies. Based on Definition 2, we know that in order to obtain the certain answer to a query over an ontology, we need to compute all four-valued models of the ontology. However, it is difficult to do that. Can we obtain the certain answer only through one model? In the following, we show that the four-valued model \( 4\text{-can}(K) \) can serve as such a model. First let us see some properties which hold for \( 4\text{-can}(K) \).

4.2.1. Properties for \( 4\text{-can}(K) \)

In order to obtain the certain answer to a query over an ontology through the four-valued model \( 4\text{-can}(K) \), we need to know the relation between \( 4\text{-can}(K) \) and
other four-valued models. The following lemma gives such a relation.

**Lemma 1** Let \( K = \langle T, A \rangle \) be a DL-Lite ontology, and \( M = \langle \Delta^M, M^\prime \rangle \) be a four-valued model for \( K \). Then, there is a function \( \psi \) from \( \Delta^\text{can}(K) \) to \( \Delta^M \) such that

1. For each atomic concept \( A \) in \( K \) and each object \( o \in \Delta^\text{can}(K) \), if \( o \in \text{proj}^+(A^\text{can}(K)) \), then \( \psi(o) \in \text{proj}^+(A^M) \).
2. For each atomic role \( P \) in \( K \) and each pair of objects \( o, o' \in \Delta^\text{can}(K) \), if \( (o, o') \in \text{proj}^+(P^\text{can}(K)) \), then \( (\psi(o), \psi(o')) \in \text{proj}^+(P^M) \).

**Proof.** This proof is analogous to the proof of Lemma 7 in [6].

From Lemma 1 we know that, for every four-valued model \( M \) of \( K = \langle T, A \rangle \), there is a homomorphism from \( 4\text{-can}(K) \) to \( M \) that maps the objects which support concepts and roles to the \( t \) or \( \overline{t} \) in \( 4\text{-can}(K) \) to objects which support the same concepts and roles to be \( t \) or \( \overline{t} \) in \( M \). We know that for a union of conjunctive queries \( q \) over an ontology \( K = \langle T, A \rangle \), a certain answer to \( q \) is only related with the constants which support concepts and roles to be \( t \) or \( \overline{t} \) in every four-valued model of \( K \). So we have the following theorem which is analogous to Theorem 29 in [6].

**Theorem 3** Let \( K = \langle T, A \rangle \) be a DL-Lite ontology, and \( Q \) be a union of conjunctive queries over \( K \). Then \( 4\text{-Ans}(Q, K) = Q^\text{4\text{-can}}(K) \).

**Proof.** This proof is analogous to the proof of Theorem 29 in [6].

Theorem 3 tells us that for any union of conjunctive queries \( Q \) the answers to \( Q \) over \( K \) correspond to the evaluation of \( Q \) in \( 4\text{-can}(K) \). So in order to obtain the certain answers to \( Q \) over \( K \), we only need to compute one four-valued model, that is, \( 4\text{-can}(K) \), instead of computing all four-valued models.

Calvanese et al. have proved that the set of answers to a union of conjunctive queries \( Q \) in DL-Lite corresponds to the union of the answers to the various conjunctive queries in \( Q \) [6]. We can show similar result for four-valued semantics.

**Theorem 4** Let \( K = \langle T, A \rangle \) be a DL-Lite ontology, and let \( Q \) be a union of conjunctive queries over \( K \). Then \( 4\text{-Ans}(Q, K) = \bigcup_{q \in Q} 4\text{-Ans}(q_i, K) \).

**Proof.** We first show that \( \bigcup_{q \in Q} 4\text{-Ans}(q_i, K) \subseteq 4\text{-Ans}(Q, K) \). Suppose that \( \overline{t} \in \bigcup_{q \in Q} 4\text{-Ans}(q_i, K) \). Then we can deduce that there must exist \( i \in \{1, \ldots, k\} \) such that \( \overline{t} \in 4\text{-Ans}(q_i, K) \). Since \( Q = \bigvee_{i=1, \ldots, k} q_i \), \( \overline{t} \in 4\text{-Ans}(Q, K) \).

Then we show \( 4\text{-Ans}(Q, K) \subseteq \bigcup_{q_i \in Q} 4\text{-Ans}(q_i, K) \). Suppose \( \overline{t} \in 4\text{-Ans}(Q, K) \), and suppose that every \( q_i \) is of the form \( q_i(\vec{x}) \leftarrow \text{conj}_j(\vec{x}, \vec{y}_j) \) for each \( q_i \in Q \). Then by Theorem 3, \( \overline{t}^\text{can}(K) = Q^\text{4\text{-can}}(K) \), which implies that there exists \( i \in \{1, \ldots, k\} \) such that \( \overline{t}^\text{can}(K) = \text{conj}_j(\vec{x}, \vec{y}_j)^\text{4\text{-can}}(K) \). Hence, from Theorem 3, it follows that \( \overline{t} \in 4\text{-Ans}(q_i, K) \).

**4.2.2. Query evaluation under four-valued semantics**

Since \( 4\text{-can}(K) \) may be infinite, it may be impossible to obtain the answer by \( 4\text{-can}(K) \). Calvanese et al. have given an algorithm, that is \( \text{PerfectRef} \), to rewrite a query based on PIs in TBox, then the reformulation of the query is not related with TBox. We now extend the results in [6] by considering generic DL-Lite ontologies (either satisfiable or unsatisfiable), and provide the following theorem which shows that query answering for UCQs over DL-Lite ontologies can also be reduced to evaluate a finite reformulation of the query over \( 4\text{-db}(A) \).

**Theorem 5** Let \( K = \langle T, A \rangle \) be an inconsistent DL-Lite ontology, \( q \) a conjunctive query over \( K \), and let \( PR \) be a union of conjunctive queries returned by \( \text{PerfectRef}(q, t) \). Then \( 4\text{-Ans}(q, K) = PR^\text{4\text{-db}}(A) \), where \( PR^\text{4\text{-db}}(A) \) is defined by Definition 2.

**Proof.** The proof is a consequence of Lemma 39 in [6] by extending the classical semantics to four-valued semantics.

Based on Theorem 5, we have the following Algorithm \( \text{Answer}(Q, K) \):

**Input**: a DL-Lite ontology \( K = \langle T, A \rangle \), a UCQ \( Q \);

**Output**: \( 4\text{-Ans}(Q, K) \);

**Return**: \( (\bigcup_{q_i \in Q} \text{PerfectRef}(q_i, T))^\text{4\text{-db}}(A) \).

In fact, Algorithm \( \text{Answer}(Q, K) \) first computes the perfect reformulation \( PR \) of \( Q \) by rewriting rule \( \text{PerfectRef} \), then returns \( PR^\text{4\text{-db}}(A) \) directly. The following theorem shows the correctness of the algorithm.

**Theorem 6** Let \( K = \langle T, A \rangle \) be a DL-Lite ontology, and \( Q \) be a union of conjunctive queries, and \( \vec{t} \) a tuple of constants in \( K \). We have: (1) the algorithm \( \text{Answer}(Q, K) \) terminates; (2) \( \vec{t} \in 4\text{-Ans}(Q, K) \) if and only if \( \vec{t} \in \text{Answer}(Q, K) \).
Proof. It is clear that (1) holds because the algorithm Answer\((Q, K)\) mainly depends on the algorithm PerfectRef\((q, T)\) which terminates [6]. Through Theorem 4 and Theorem 5 we can easily obtain (2). □

Example 4 (Example 2 contd.) Let us consider a query \(q(x) \leftarrow \text{Stud}(x) \land \exists y.\text{hasTutor}(x, y)\). According to Answer\((q, K)\), it first executes PerfectRef\((q, T)\) as follows: First PerfectRef\((q, T)\) inserts a new query \(q(x) \leftarrow \text{PhDStud}(x) \land \exists y.\text{hasTutor}(x, y)\) by applying the atom Stud\((x)\) to the PI PhDStud \(\subseteq\) Stud. Then the query \(q(x) \leftarrow \text{PhDStud}(x) \land \text{Stud}(x)\) is added by applying the atom \(\exists y.\text{hasTutor}(x, y)\) to the PI Stud \(\subseteq\) hasTutor. Next the query \(q(x) \leftarrow \text{PhDStud}(x)\) is added according to application of the PI PhDStud \(\subseteq\) Stud to the atom Stud\((x)\). So the returned result of PerfectRef\((q, T)\) is the union of the following conjunctive queries:

\[
\{ q(x) \leftarrow \text{Stud}(x) \land \exists y.\text{hasTutor}(x, y), q(x) \leftarrow \text{PhDStud}(x) \land \exists y.\text{hasTutor}(x, y), q(x) \leftarrow \text{PhDStud}(x) \land \text{Stud}(x), q(x) \leftarrow \text{PhDStud}(x) \}\.
\]

Then based on the interpretation 4-db\((A)\), we obtain that 4-\(\text{Ans}(q, K)\) is \(\{ a \}\).

The following theorem shows the complexity of paraconsistent query answering over DL-Lite ontologies:

Theorem 7 Paraconsistent answering unions of conjunctive queries in DL-Lite \(K = (T, A)\) is PTime in the size of the TBox, and LOGSPACE in the size of the ABox.

Proof. Based on the algorithm Answer\((Q, K)\), we know that the complexity of paraconsistent query answering over DL-Lite ontologies mainly depends on the algorithm PerfectRef\((q, T)\) and the evaluation of a union of conjunctive queries over 4-db\((A)\). First the algorithm PerfectRef\((q, T)\) runs in time polynomial in the size of \(T\) [6] which return unions of conjunctive queries. Since unions of conjunctive queries are a subclass of FOL queries [6] and the answer set is only related to the constants in \(\Delta^{4\text{db} (A)}\), which support concepts or roles to be \(\top\) in the interpretation 4-db\((A)\), so the union of conjunctive queries can be directly evaluated by an SQL engine over the ABox. The LOGSPACE complexity is achieved by the fact that the evaluation of unions of conjunctive queries over a database can be computed in LOGSPACE with respect to the size of the database.

The process of our method is described in Figure 2. That is, given a query \(Q\) and a DL-Lite ontology \(O = (T, A)\), our method first rewrites the query \(Q\) and obtains the reformulation of the query, that is PR, and then computes 4-db\((A)\), finally, through PR and 4-db\((A)\), we obtain the query result.

4.3. Implementation

Our algorithm has been implemented in Java based on the Owlgres reasoner\(^1\). The following sample illustrates the nontrivial results of our algorithm for consistent query answering over an inconsistent ontologies: the bird ontology which are reported in [9]. For this ontology, we give two queries, the queries and query results are as follows:

\[
\begin{align*}
q_1(x) & \leftarrow \text{fly}(x), \text{penguin}(x) \\
q_2(x) & \leftarrow \text{fly}(x), \text{Animal}(x)
\end{align*}
\]

query result: tweety

query result: fred, tweety

For \(q_1(x), q_2(x)\), we may obtain all possible results if we use classical reason because there exist two conflict atoms such as “penguin \(\subseteq\) fly, penguin \(\subseteq\) ¬fly” in bird ontology. However, we obtain the answer set with “tweety” for \(q_1(x)\) and the answer set with “tweety, fred” for \(q_2(x)\) through our method.

5. Comparison and evaluation

5.1. Comparison with the reduction method

Ma et al. [13,14,12] have proposed a method for reasoning with inconsistent ontologies through reducing an inconsistent ontology under four-valued semantics to a consistent one under classical semantics. Although they have not considered CQA, we can apply the re-

\(^1\)http://clarkparsia.com/weblog/2008/03/23/owlgres-scalable-db/
duction method to an inconsistent ontology and then do query answering over the resulting ontology. Figure 3 gives the process of using the reduction approach to query answering over DL-Lite ontologies. For simplicity, we call this approach ReductMethod.

From Figure 2 and Figure 3, we can find that our method is the same as ReductMethod except that ReductMethod need to reduce inconsistent ontologies and queries under four-valued semantics to consistent ontologies and queries under classical semantics. In fact, for a query and an inconsistent DL-Lite ontology, the result obtained by our method is the same as the result obtained by ReductMethod. See the following theorem.

**Theorem 8** Let $\mathcal{K} = \langle T, A \rangle$ be a DL-Lite ontology, $Q$ be a conjunctive query, $\pi$ be the reduction function given in [11], then we have $\text{PerfectRef}(Q, T)^{\text{4-db}(A)} = \text{PerfectRef}(\pi(Q), \pi(T))^{\text{db}(\pi(A))}$. \hfill $\square$

**Proof.** First, based on [11], we know that when we apply the reduction method, an atomic concept $A$ (resp. an atomic role $P$) is not changed, that is, $\pi(A) = A$ and $\pi(P) = P$. Since ABox $A$ is formed by a finite set of membership assertions on atomic concepts and on atomic roles, $\pi(A) = A$. Since for any query $Q$, atoms of $Q$ are of the form $A(z)$ or $P(z_1, z_2)$, where $A$ and $P$ are respectively an atomic concept and an atomic role of $K$, and $z, z_1, z_2$ are either constants in $K$ or variables, based on [11], we have $Q = \pi(Q)$. Similarly, PIs in $T$ are not changed after reduction. Furthermore, for each NI in $T$ (excluding those NIs whose right sides is of the form $\neg \exists R$, after reduction, it will be changed to a new PI whose left side is the same as the left side of the NI and whose right side is a new concept which does not occur in $T$. For those NIs whose right sides are of the form $\exists R$, after reduction, they will be changed to new NIs which do not occur in $T$.

Based on [6], we know that the query rewriting function PerfectRef is only related with PIs. Moreover, those PIs which are in $\pi(T)$ but not in $T$ can not affects the result of PerfectRef because their right sides are new concepts or roles which do not occur in $K$. We can obtain that $\text{PerfectRef}(\pi(Q), \pi(T)) = \text{PerfectRef}(Q, T)$.

Through the algorithm Answer$(Q, K)$ and four-valued semantics, we know that the query result is related with $\text{proj}^+(A^{\text{4-db}(A)})$ (resp., $\text{proj}^+(P^{\text{4-db}(A)})$) when using our method. Also, the query result is related with $A^{\text{db}(A)}$ (resp., $P^{\text{db}(A)}$) when using the ReductMethod. By the definition of 4-db($A$), we know that for any atomic concept $A$ and atomic role $P$ in $A$, $A^{\text{db}(A)} = \text{proj}^+(A^{\text{4-db}(A)})$ and $P^{\text{db}(A)} = \text{proj}^+(P^{\text{4-db}(A)})$ respectively. From the above discussions, it is clear $\text{PerfectRef}(Q, T)^{\text{4-db}(A)} = \text{PerfectRef}(\pi(Q), \pi(T))^{\text{db}(\pi(A))}$. So the claim holds.

Theorem 8 shows that for a given query over a DL-Lite ontology, both our method and the ReductMethod return the same set of answers. However, our method has the following advantages over the ReductMethod:

- First, from the proof of Theorem 8, $\pi(T)$ have more PIs than $T$, so it takes more time to compute $\text{PerfectRef}(\pi(Q), \pi(T))$ than to compute $\text{PerfectRef}(Q, T)$;
- Second, it takes time to reduce inconsistent ontologies and the queries by applying the ReductMethod. Although the complexity of the reduction has linear complexity, it can take a lot of time for large ontologies.
- After the users obtain the answers to a conjunctive query over an inconsistent ontology, it is very likely that they want to know the explanation of the answers. With our approach, it is easy to adapt the algorithm for the explanation of an answer to a conjunctive query over a DL-Lite ontology in [4] to provide such an explanation. However, with the reduction approach, it is difficult to provide such an explanation facility.

### 5.2. Evaluation

The main goal of the evaluation is to compare the running time of query rewriting when using $T$ and
\[ \pi(T) \]. We revised ParOWL\(^2\) to deal with DL-Lite ontologies. Then we use PerfectRef\(^3\) to compute the query reformulation. The experiments were performed on a Windows 7 system and 1024MB maximal heap space was set. The data sets are proton\(^4\). Table 3 shows some characteristics of the data sets used for our experiments.

<table>
<thead>
<tr>
<th>Ontology</th>
<th>PIs</th>
<th>NIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>proton(_{100}_all)</td>
<td>327</td>
<td>616</td>
</tr>
<tr>
<td>proton(_{100}_rudis)</td>
<td>327</td>
<td>931</td>
</tr>
<tr>
<td>proton(_{100}_studis)</td>
<td>327</td>
<td>846</td>
</tr>
<tr>
<td>proton(_{50}_all)</td>
<td>327</td>
<td>1276</td>
</tr>
<tr>
<td>proton(_{50}_rudis)</td>
<td>327</td>
<td>1297</td>
</tr>
<tr>
<td>proton(_{50}_studis)</td>
<td>327</td>
<td>1346</td>
</tr>
</tbody>
</table>

Then we give 4 queries with different size as follows:

- \(Q_1(x) \leftarrow \text{Organization}(x) \land \text{dointBusinessAs}(x,y)\)
- \(Q_2(x) \leftarrow \text{Organization}(x) \land \text{Group}(x) \land \text{dointBusinessAs}(x,y)\)
- \(Q_3(x) \leftarrow \text{Organization}(x) \land \text{Group}(x) \land \text{dointBusinessAs}(x,y) \land \text{Alias}(y)\)
- \(Q_4(x) \leftarrow \text{Organization}(x) \land \text{Group}(x) \land \text{dointBusinessAs}(x,y) \land \text{Alias}(y) \land \text{hasAlias}(z,y)\)

Table 4 shows the experimental results. From Table 4, we can find that the running time of query rewriting is related with the number of PIs and the number of atoms and terms occurring in the query. For each DL-Lite TBox \(T\) and a query \(Q\) over it, if we let \(\pi(T)\) be the reduction TBox, then the running time to compute \(\text{PerfectRef}(Q, \pi(T))\) is more than that to compute \(\text{PerfectRef}(Q, T)\).

### 6. Related work

We discuss related work on consistent query answering over logic-based knowledge bases in this section. For consistent query answering over inconsistent databases or inconsistent propositional knowledge bases, some approaches are known \[17, 1, 5, 10, 8, 7\]. The approaches in \[1, 5\] mainly rely on the notion of repair for a database instance that may violate integrity constraints specified over its schema. However consistent query answering of conjunctive queries expressed over database schemas with integrity constraints is a coNP-complete problem in data complexity, i.e., the complexity measured only with respect to the size of the database instance \[1\]. Villadsen \[17\] has proposed a many-valued paraconsistent logic for query answering based on the notion of indeterminacy, which only considers instance checking.

Huang et al. \[8, 7\] have proposed an approach to reasoning with inconsistent ontologies. However, their approach is based on consistent sub-ontologies obtained through selection function, which will lead to some information loss. Moreover, only instance checking is discussed instead of conjunctive queries as done in this paper.

Ma et al. \[13, 14, 12\] have proposed a method for reasoning with inconsistent ontologies through reducing an inconsistent ontology under four-valued semantics to a consistent one under classical semantics. However, for a DL-Lite ontology, more PIs will be added after reduction, thus lead to more time for query rewriting. Besides, the polynomial reduction will be an extra burden for the PTime query answering over DL-Lite.

Lembo et al. \[10\] have proposed an approach for consistent instance checking based on an inconsistency-tolerant semantics. Their method relies on the notion of repair by deleting some inconsistent membership assertions, which again will lead to information loss. Furthermore, they have shown that their approach to consistent conjunctive query answering over inconsistent DL-Lite ontologies is general intractable with respect to data complexity.

### 7. Conclusion and future work

In this paper, we have investigated how to deal with the problem of consistent query answering for conjunctive queries posed to inconsistent DL-Lite ontologies. Unlike existing approaches, we consider four-valued semantics and proposed a tractable approach to paraconsistent query answering over DL-Lite ontologies. Our approach can guarantee LogSpace complexity with respect to the size of ABox.

For future work, we will study the problem of paraconsistent query answering over more expressive DL-Lite such as DL-Lite\(_R\) and DL-Lite\(_F\).

---

\(^2\)http://logic.aifb.uni-karlsruhe.de/wiki/Paraconsistent reasoning
\(^3\)http://www.comlab.ox.ac.uk/projects/requiem/
\(^4\)http://radon.ontoware.org/downloads.htm
Acknowledgments

We thank the anonymous reviewers for their valuable suggestions and comments. We also want to thank Jianfeng Du for his valuable comments on the draft paper and Chang Liu for his help on programming. Our research is supported by the National Grand Fundamental Research 973 Program of China under grant (No.2007CB307100, No.2007CB307106) and by the Specialized Research Foundation of Doctoral Program of Higher Education of China under grant (No.20050004008).

References


