Rewriting-Based Instance Retrieval for Negated Concepts in Description Logic Ontologies

Jianfeng Du[1(✉)] and Jeff Z. Pan[2]

[1] Guangdong University of Foreign Studies, Guangzhou 510006, China
jfdu@gdufs.edu.cn
[2] The University of Aberdeen, Aberdeen AB24 3UE, UK

Abstract. Instance retrieval computes all instances of a given concept in a consistent description logic (DL) ontology. Although it is a popular task for ontology reasoning, there is no scalable method for instance retrieval for negated concepts by now. This paper studies a new approach to instance retrieval for negated concepts based on query rewriting. A class of DL ontologies called the *inconsistency-based first-order rewritable* (*IFO-rewritable*) class is identified. This class guarantees that instance retrieval for an atomic negation can be reduced to answering a disjunction of conjunctive queries (CQs) over the ABox. The IFO-rewritable class is more expressive than the first-order rewritable class which guarantees that answering a CQ is reducible to answering a disjunction of CQs over the ABox regardless of the TBox. Two sufficient conditions are proposed to detect IFO-rewritable ontologies that are not first-order rewritable. A rewriting-based method for retrieving instances of a negated concept is proposed for IFO-rewritable ontologies. Preliminary experimental results on retrieving instances of all atomic negations show that this method is significantly more efficient than existing methods implemented in state-of-the-art DL systems.

1 Introduction

Description logics (DLs) [2] are popular knowledge representation languages underpinning the Web Ontology Language (OWL). A DL ontology consists of a TBox and an ABox, where the TBox describes relations between concepts and roles, and the ABox describes instances of concepts and roles. DLs enable a number of tasks for ontology reasoning based on the classical first-order semantics. Among these tasks, instance retrieval is a popular one which computes all individuals in a consistent DL ontology that are instances of a given concept.

Most studies on instance retrieval focus on atomic concepts, namely concept names. It is well-known that instance retrieval for atomic concepts is tractable for those DLs that underpin the three profiles QL, EL and RL of OWL 2, the newest version of OWL. There have also been optimization and approximation techniques proposed for instance retrieval in expressive DLs [13,17,21]. However, to the best of our knowledge, there is seldom any dedicated study on instance retrieval for negated concepts. A negated concept is of the form $\neg C$ where C is

© Springer International Publishing Switzerland 2015
M. Arenas et al. (Eds.): ISWC 2015, Part I, LNCS 9366, pp. 339–355, 2015.
DOI: 10.1007/978-3-319-25007-6_20

a DL concept without the negation symbol $\neg$. In the reality, it is often required to compute instances of a negated concept. For example, one may often raise questions like the following ones upon a DL ontology describing people in universities: Who is not an undergraduate? Who does not have a friend who is a professor? Who does not get a PhD degree from a Chinese university?

According to the semantics of DLs, the set of instances of a negated concept $\neg C$ cannot be returned as the set of those instances explicitly declared in the given ontology, nor the complement set of the set of instances of C. A common sound and complete method for instance retrieval for $\neg C$ is to reduce the problem to instance retrieval for P_C, where P_C is a fresh atomic concept, by adding an axiom $\neg C \sqsubseteq P_C$ to the TBox. This method is commonly implemented in state-of-the-art DL systems. However, the added axiom $\neg C \sqsubseteq P_C$ introduces concept disjunctions and cannot be expressed in DLs that guarantee tractable instance retrieval. To guarantee tractability, one may apply another method which retrieves all instances of $\neg C$ by checking if $\{C(a)\}$ is consistent with the given ontology for every individual a in the ontology. However, this method is hardly scalable for large DL ontologies with many individuals.

Inspired by the successful query rewriting approach to ontology reasoning (see e.g. [5,7]), we solve the problem of instance retrieval for negated concepts from a new perspective, i.e., by query rewriting. There are some challenges in making use of query rewriting. First of all, existing query rewriting methods are designed for conjunctive queries (CQs) but not for negated concepts. It requires us to establish a connection between negated concepts and CQs. More importantly, it is required that the applicable DLs be as expressive as possible.

In this paper we tackle all the above challenges and make the following contributions. Firstly, we identify a class of DL ontologies, called the *inconsistency-based first-order rewritable* (*IFO-rewritable*) class, which guarantees that instance retrieval for an atomic negation is reducible to answering a disjunction of CQs over the ABox. The class is characterized by an *inconsistency-rewritten set* of Boolean conjunctive queries (BCQs) whose size is finite and independent of the ABox. From an inconsistency-rewritten set $\mathcal{S}$ of BCQs for an atomic concept A, we can extract a disjunction of CQs $Q_D(x)$ in time linear in the size of $\mathcal{S}$ such that the set of instances of $\neg A$ is the set of answers of $Q_D(x)$ in the ABox.

Secondly, we show that the IFO-rewritable class is more expressive than the first-order rewritable class, where the latter is commonly used in query rewriting and guarantees that answering a CQ is reducible to answering a disjunction of CQs over the ABox. We propose two sufficient conditions for detecting IFO-rewritable ontologies that are not first-order rewritable. One condition relies on extracting first-order rewritable subsets of the given ontology and can be checked regardless of the ABox in time polynomial in the size of the TBox. The other condition is applicable to the DL $\mathcal{ELR}_\perp$ and can be checked in time polynomial in the size of the ABox by considering certain subgraphs of the ABox graph.

Finally, we conduct experiments on large first-order rewritable ontologies to demonstrate the advantages of exploiting inconsistency-rewritten sets of BCQs to retrieve instances of atomic negations. We compare this proposed method with the two aforementioned methods for instance retrieval for atomic negations. Experimental results obtained by several state-of-the-art DL systems show that the proposed method is significantly more efficient than the two aforementioned methods in retrieving instances of all atomic negations. Moreover, among all compared methods only the proposed one is scalable for large ontologies with tens of millions of assertions. All proofs are available at http://www.dataminingcenter.net/rebsir/ISWC15-full.pdf.

2 Preliminaries

We assume that the reader is familiar with DLs [2]. A DL ontology consists of a TBox and an ABox, where the TBox is a finite set of axioms on relations between concepts and roles, and the ABox is a finite set of assertions declaring instances of concepts and roles. In this work we only consider *normalized* ABoxes. An ABox is said to be *normalized* if it consists of *basic assertions* that are concept assertions of the form $A(a)$ or role assertions of the form $r(a, b)$, where A is an atomic concept, r is an atomic role, and a and b are individuals. Other concept assertions and role assertions can be normalized to basic ones in a standard way. For a normalized ABox $\mathcal{A}$, let $\mathsf{Ind}(\mathcal{A})$ denote the set of individuals appearing $\mathcal{A}$ and $\mathsf{Cn}(\mathcal{A})$ the set of atomic concepts appearing in $\mathcal{A}$.

The semantics of DLs coincides with the classical first-order semantics. A DL ontology $\mathcal{O}$ is said to be *consistent*, denoted by $\mathcal{O} \not\models \bot$, if it has at least one model, otherwise *inconsistent*, denoted by $\mathcal{O} \models \bot$. An ABox $\mathcal{A}$ is said to be *consistent with* a TBox $\mathcal{T}$ if $\mathcal{T} \cup \mathcal{A}$ is consistent. An individual a is said to be an *instance* of a concept C in $\mathcal{O}$ if the concept assertion $C(a)$ is satisfied by all models of $\mathcal{O}$, denoted by $\mathcal{O} \models C(a)$. The problem of *instance retrieval* for C in $\mathcal{O}$ is to compute the set of instances of C in $\mathcal{O}$.

A *conjunctive query* (*CQ*) $Q(\boldsymbol{x})$ is a formula of the form $\exists \boldsymbol{y}\, \phi(\boldsymbol{x}, \boldsymbol{y})$, where $\phi(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms over atomic concepts, atomic roles and the (in)equality predicate, $\boldsymbol{x}$ are *answer variables*, and $\boldsymbol{y}$ are *quantified variables*. A CQ without answer variables is called a *Boolean conjunctive query* (*BCQ*). Here a BCQ is written and treated as a set of atoms. For example, the BCQ $\exists x\, A(x) \wedge B(x)$ is written as $\{A(x), B(x)\}$. A *disjunction* of CQs, also called a *union* of CQs or a *UCQ* in the literature, is a formula of the form $Q_1(\boldsymbol{x}) \vee \ldots \vee Q_n(\boldsymbol{x})$ where $n \geq 1$ and $Q_1(\boldsymbol{x}), \ldots, Q_n(\boldsymbol{x})$ are CQs. We say a disjunction of BCQs Q_D is *entailed by* $\mathcal{O}$, denoted by $\mathcal{O} \models Q_D$, if Q_D is satisfied by all models of $\mathcal{O}$. A tuple $\boldsymbol{t}$ of individuals is called an *answer* to a disjunction of CQs $Q_D(\boldsymbol{x})$ in an ontology $\mathcal{O}$ if $\mathcal{O} \models Q_D(\boldsymbol{t})$, where $Q_D(\boldsymbol{t})$ is a disjunction of BCQs obtained from $Q(\boldsymbol{x})$ by replacing variables in $\boldsymbol{x}$ with corresponding individuals in $\boldsymbol{t}$. The set of answers to $Q_D(\boldsymbol{x})$ in $\mathcal{O}$ is denoted by $\mathsf{ans}(\mathcal{O}, Q_D(\boldsymbol{x}))$. For a disjunction of BCQs Q_D, $\mathsf{ans}(\mathcal{O}, Q_D) = \{\langle\rangle\}$ if $\mathcal{O} \models Q_D$, or $\mathsf{ans}(\mathcal{O}, Q_D) = \emptyset$ otherwise.

Datalog$^{\pm}$ [4] is highly related to DLs. It extends datalog with existential rules, which are formulae of the form $\forall \boldsymbol{x} \forall \boldsymbol{y}\, \phi(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \exists \boldsymbol{z}\, \varphi(\boldsymbol{x}, \boldsymbol{z})$, where $\phi(\boldsymbol{x}, \boldsymbol{y})$ and $\varphi(\boldsymbol{x}, \boldsymbol{z})$ are conjunctions of atoms (often treated as sets of atoms), and $\boldsymbol{x}$,

$\boldsymbol{y}$ and $\boldsymbol{z}$ are pairwise disjoint sets of variables. The part of R at left-hand side of $\rightarrow$ is the *body* of R, whereas the part of R at right-hand side of $\rightarrow$ is the *head* of R. An existential rule R is called an *equality generating dependency* (*EGD*) if the head of R is of the form $x_1 = x_2$ where x_1 and x_2 are different variables appearing in the body of R; called a *constraint* if the head of R is empty; otherwise, called a *tuple generating dependency* (*TGD*). A TGD is said to be *linear* if its body contains a single atom; *multi-linear* if all atoms in its body have the same variables. A linear TGD is also a multi-linear TGD.

A datalog$^{\pm}$ program is a finite set of existential rules, amounting to the conjunction of all existential rules in it. Since existential rules are formulae in first-order logic with equality, a TBox expressed in some DLs can be translated to a datalog$^{\pm}$ program. It follows that an ontology expressed in some DLs can be translated to the union of a datalog$^{\pm}$ program and a normalized ABox. We call such an ontology *datalog$^{\pm}$-translatable*. For a datalog$^{\pm}$-translatable ontology with TBox $\mathcal{T}$, throughout this paper we use $\mathcal{S}^D_{\mathcal{T}}$ to denote the set of TGDs translated from $\mathcal{T}$, $\mathcal{S}^C_{\mathcal{T}}$ to denote the set of constraints translated from $\mathcal{T}$, and $\mathcal{S}^E_{\mathcal{T}}$ to denote the set of EGDs translated from $\mathcal{T}$. Since datalog$^{\pm}$ works with the *unique name assumption*, this assumption is also adopted in an arbitrary datalog$^{\pm}$-translatable ontology, which means that all individuals appearing in the ontology are interpreted as different in any model of the ontology.

By $|S|$ we denote the cardinality of a set S. A *substitution* for a first-order entity (such as atom, formula, etc.) E is a mapping from variables in E to individuals or variables; it is *ground* if it maps variables in E to individuals only.

We recall the notions of first-order rewritability and separability in the context of DLs [4]. A set $\mathcal{S}^D$ of TGDs is said to be *first-order rewritable* if, for every conjunctive query $Q(\boldsymbol{x})$, there exists a finite disjunction of conjunctive queries $Q_D(\boldsymbol{x})$ such that $\mathsf{ans}(\mathcal{S}^D \cup \mathcal{A}, Q(\boldsymbol{x})) = \mathsf{ans}(\mathcal{A}, Q_D(\boldsymbol{x}))$ for all ABoxes $\mathcal{A}$. It has been shown [4] that a set $\mathcal{S}^D$ of TGDs is first-order rewritable if all TGDs in $\mathcal{S}^D$ are multi-linear. A set $\mathcal{S}^E$ of EGDs is said to be *separable* from a set $\mathcal{S}^D$ of TGDs if the following holds for every ABox $\mathcal{A}$: if there exists an EGD $\forall \boldsymbol{x}\, \phi(\boldsymbol{x}) \rightarrow x_1 = x_2$ in $\mathcal{S}^E$ and a ground substitution σ for $\boldsymbol{x}$ such that $\mathcal{S}^D \cup \mathcal{A} \models \phi(\boldsymbol{x}\sigma)$ and $x_1\sigma \neq x_2\sigma$, then there is a ground substitution θ for $\boldsymbol{x}$ such that $\mathcal{A} \models \phi(\boldsymbol{x}\theta)$ and $x_1\theta \neq x_2\theta$; otherwise, $\mathcal{S}^D \cup \mathcal{S}^E \cup \mathcal{A} \models Q$ if and only if $\mathcal{S}^D \cup \mathcal{A} \models Q$ for all BCQs Q. It has been shown [4] that deciding if $\mathcal{S}^D \cup \mathcal{S}^C \cup \mathcal{S}^E \cup \mathcal{A} \models Q$ for a first-order rewritable set $\mathcal{S}^D$ of TGDs, a set $\mathcal{S}^C$ of constraints, a set $\mathcal{S}^E$ of EGDs separable from $\mathcal{S}^D$, an ABox $\mathcal{A}$ and a BCQ Q is in AC^0 in *data complexity*, the complexity measured in the size of $\mathcal{A}$ only.

3 Rewriting-Based Instance Retrieval

Query rewriting is an efficient and scalable approach to reasoning over ontologies that are expressed in lightweight DLs and have large ABoxes. The idea is to rewrite a given CQ into a disjunction of CQs or a datalog program such that the given CQ can be answering by evaluating the rewriting result over the ABox only. Query rewriting has been implemented in modern DL systems such as

Rapid [7] and MASTRO [5]. To enable query rewriting for all CQs, we require that the back-end ontology be *first-order rewritable* based on the first-order rewritable class of TGDs. We call a DL ontology $\mathcal{O}$ with TBox $\mathcal{T}$ *first-order rewritable* if it is a datalog$^{\pm}$-translatable ontology such that $\mathcal{S}_{\mathcal{T}}^{D}$ is first-order rewritable, $\mathcal{S}_{\mathcal{T}}^{E}$ is separable from $\mathcal{S}_{\mathcal{T}}^{D}$ and $\mathcal{S}_{\mathcal{T}}^{C}$ can be arbitrary. It has been shown [4] that most DLs in the DL-Lite family [6], such as DL-Lite$_X$ and DL-Lite$_{X,\sqcap}$ for $X \in \{\mathcal{F}, \mathcal{R}, \mathcal{A}\}$, express first-order rewritable ontologies, where DL-Lite$_{\mathcal{R}}$ underpins the QL profile of OWL 2.

Inspired by the exciting progress on query rewriting, we intend to solve the problem of instance retrieval for negated concepts by exploiting query rewriting. As mentioned in Section 1, there are challenges in making use of query rewriting, including the establishment of a bridge from negated concepts to CQs and the guarantee of the expressivity for applicable DLs. To tackle these challenges, we start with the study on instance retrieval for atomic negations.

Given a consistent DL ontology $\mathcal{O}$ with ABox $\mathcal{A}$ and an atomic negation $\neg A$, the basic idea for instance retrieval for $\neg A$ in $\mathcal{O}$ is to compute a disjunction of CQs $Q_D(x)$ such that the set of answers to $Q_D(x)$ in $\mathcal{A}$ amounts to the set of instances of $\neg A$ in $\mathcal{O}$. To compute $Q_D(x)$, we first compute a set $\mathcal{S}$ of BCQs, each of which is entailed by $\mathcal{A} \cup \{A(a)\}$ for some instance a of $\neg A$ in $\mathcal{O}$, and then construct $Q_D(x)$ from $\mathcal{S}$. We call the set of BCQs computed in the first step an *inconsistency-rewritten set*, which is formally defined below.

Definition 1. *Given an atomic concept A and a consistent DL ontology $\mathcal{O}$ with ABox $\mathcal{A}$, a set $\mathcal{S}$ of BCQs is called an* inconsistency-rewritten set *for A in $\mathcal{O}$ if $\mathcal{S}$ has a finite size that is independent of the size of $\mathcal{A}$, $\mathcal{A} \not\models \bigvee \mathcal{S}$, and for all individuals a in $\mathcal{O}$, $\mathcal{O} \cup \{A(a)\} \models \bot$ if and only if $\mathcal{A} \cup \{A(a)\} \models \bigvee \mathcal{S}$. $\mathcal{O}$ is said to be* inconsistency-based first-order rewritable *(simply* IFO-rewritable*) for A, if there is an inconsistency-rewritten set of BCQs for A in $\mathcal{O}$.*

The IFO-rewritable class is characterized by inconsistency-rewritten sets of BCQs. Note that all BCQs considered in this paper do not contain individuals. After an inconsistency-rewritten set $\mathcal{S}$ of BCQs for A is computed, the set of instances of $\neg A$ in an IFO-rewritable ontology $\mathcal{O}$ for A can be computed as the set of answers to a disjunction of CQs $Q_D(x)$ in the ABox $\mathcal{A}$ of $\mathcal{O}$, where $Q_D(x)$ is extracted from $\mathcal{S}$ by separating an atom over A from the other atoms in every BCQ in $\mathcal{S}$. We introduce some notations to explain this method. Given a set $\mathcal{S}$ of BCQs and an atomic concept A, we call the set of BCQs in $\mathcal{S}$ that contain at least one atom over A the *A-projection* of $\mathcal{S}$, denoted by $\mathcal{S}|_A$. For a BCQ Q that has some atoms over A, we call a pair $\langle A(t), Q' \rangle$ for t a variable an *A-bipartition* of Q if $Q' \cup \{A(t)\} = Q$ and $A(t) \notin Q'$. By $\mathsf{bipart}(Q, A)$ we denote the set of A-bipartitions of Q. Let $\varrho_{\mathcal{S},A}(z) = \bigvee\{Q'[t/z] \mid Q \in \mathcal{S}|_A, \langle A(t), Q' \rangle \in \mathsf{bipart}(Q, A)\}$, where $Q'[t/z]$ denotes the CQ obtained from Q' by renaming t to z, and z does not occur in $\mathcal{S}$ and is the unique answer variable in $Q'[t/z]$. The following theorem shows a method for extracting a disjunction of CQs from an inconsistency-rewritten set of BCQs.

Theorem 1. *Let A be an atomic concept, $\mathcal{O}$ an IFO-rewritable ontology for A with TBox $\mathcal{T}$ and ABox $\mathcal{A}$, and $\mathcal{S}$ an inconsistency-rewritten set of BCQs for A in $\mathcal{O}$. Then the set of instances of $\neg A$ in $\mathcal{O}$ is $\mathsf{ans}(\mathcal{A}, \varrho_{\mathcal{S},A}(z))$.*

Note that the disjunction of CQs computed by the above method is independent of the size of the ABox. It follows that checking an instance of a given atomic negation in an IFO-rewritable ontology can be done in AC^0 in data complexity.

Example 1. This example illustrates how to retrieve instances of an atomic negation $\neg A$ in an IFO-rewritable ontology for A after an inconsistency-rewritten set of BCQs for A is computed. Let $\mathcal{O}$ be a consistent DL ontology with TBox $\mathcal{T}$ and ABox $\mathcal{A}$. The TBox $\mathcal{T}$ consists of the following three axioms.

$$\mathsf{Husband} \sqsubseteq \exists\mathsf{marries}.\mathsf{Woman} \qquad \top \sqsubseteq \leq_1 \mathsf{marries}.\top \qquad \exists\mathsf{marries}.\mathsf{Woman} \sqsubseteq \neg\mathsf{Woman}$$

The ABox $\mathcal{A}$ consists of the following four assertions.

$$\mathsf{marries}(\mathsf{Tom},\mathsf{Ann}) \qquad \mathsf{Woman}(\mathsf{Ann}) \qquad \mathsf{marries}(\mathsf{Aba},\mathsf{Bob}) \qquad \mathsf{Woman}(\mathsf{Aba})$$

Consider computing all instances of $\neg\mathsf{Woman}$ in $\mathcal{O}$. By Definition 1 it can be checked that $\mathcal{S} = \{Q_1, Q_2, Q_3\}$ is an inconsistency-rewritten set of BCQs for Woman in $\mathcal{O}$, where $Q_1 = \{\mathsf{marries}(x,y),\ \mathsf{Woman}(y),\ \mathsf{Woman}(x)\}$, $Q_2 = \{\mathsf{Husband}(x),\ \mathsf{Woman}(x)\}$, and $Q_3 = \{\mathsf{marries}(x,y_1),\ \mathsf{marries}(x,y_2),\ y_1 \neq y_2\}$. We have $\mathcal{S}|_{\mathsf{Woman}} = \{Q_1, Q_2\}$. The disjunction of CQs extracted from $\mathcal{S}|_{\mathsf{Woman}}$ is $\varrho_{\mathcal{S},\mathsf{Woman}}(z) = (\exists y\, \mathsf{marries}(z,y) \wedge \mathsf{Woman}(y)) \vee (\exists x\, \mathsf{marries}(x,z) \wedge \mathsf{Woman}(x)) \vee \mathsf{Husband}(z)$. The set of answers to $\varrho_{\mathcal{S},\mathsf{Woman}}(z)$ in $\mathcal{A}$ is $S = \{\mathsf{Tom},\ \mathsf{Bob}\}$, which is the set of instances of $\neg\mathsf{Woman}$ in $\mathcal{O}$ by Theorem 1.

By $\rho(R)$ we denote the BCQ $\exists\boldsymbol{x}\,\phi(\boldsymbol{x})$ if R is a constraint $\forall\boldsymbol{x}\,\phi(\boldsymbol{x}) \rightarrow$, or the BCQ $\exists\boldsymbol{x}\,\phi(\boldsymbol{x}) \wedge x_1 \neq x_2$ if R is an EGD $\forall\boldsymbol{x}\,\phi(\boldsymbol{x}) \rightarrow x_1 = x_2$. For a first-order rewritable set $\mathcal{S}$ of TGDs and a BCQ Q, by $\gamma(Q,\mathcal{S})$ we denote a set $\mathcal{S}_Q$ of BCQs such that $\mathcal{S} \cup \mathcal{A} \models Q$ if and only if $\mathcal{A} \models \bigvee \mathcal{S}_Q$ for all ABoxes $\mathcal{A}$. The following theorem shows that a consistent first-order rewritable ontology is an IFO-rewritable ontology for an arbitrary atomic concept in the ontology.

Theorem 2. *Let $\mathcal{O}$ be a consistent first-order rewritable ontology with TBox $\mathcal{T}$. Then $\bigcup\{\gamma(\rho(R), \mathcal{S}^D_{\mathcal{T}}) \mid R \in \mathcal{S}^C_{\mathcal{T}}\} \cup \{\rho(R) \mid R \in \mathcal{S}^E_{\mathcal{T}}\}$ is an inconsistency-rewritten set of BCQs for an arbitrary atomic concept in $\mathcal{O}$.*

Example 2. This example illustrates how to compute an inconsistency-rewritten set of BCQs from the ontology $\mathcal{O}$ given in Example 1, which is also a first-order rewritable ontology. According to Theorem 2, we first translate $\mathcal{T}$ to the union of a set $\mathcal{S}^D_{\mathcal{T}}$ of TGDs, a set $\mathcal{S}^C_{\mathcal{T}}$ of constraints and a set $\mathcal{S}^E_{\mathcal{T}}$ of EGDs, where

$$\begin{aligned}
\mathcal{S}^D_{\mathcal{T}} &= \{\forall x\, \mathsf{Husband}(x) \rightarrow \exists y\, \mathsf{marries}(x,y) \wedge \mathsf{Woman}(y)\},\\
\mathcal{S}^C_{\mathcal{T}} &= \{\forall x,y\, \mathsf{marries}(x,y) \wedge \mathsf{Woman}(y) \wedge \mathsf{Woman}(x) \rightarrow\},\\
\mathcal{S}^E_{\mathcal{T}} &= \{\forall x,y_1,y_2\, \mathsf{marries}(x,y_1) \wedge \mathsf{marries}(x,y_2) \rightarrow y_1 = y_2\}.
\end{aligned}$$

$\mathcal{S}_{\mathcal{T}}^{D}$ contains a linear TGD and is first-order rewritable. $\mathcal{S}_{\mathcal{T}}^{E}$ is separable from $\mathcal{S}_{\mathcal{T}}^{D}$. Now, we compute $\gamma(\rho(R), \mathcal{S}_{\mathcal{T}}^{D})$ for all $R \in \mathcal{S}_{\mathcal{T}}^{C}$ by a query rewriting method such as the one implemented in Rapid [7], yielding $Q_1 = \{\mathsf{marries}(x, y), \mathsf{Woman}(y), \mathsf{Woman}(x)\}$ and $Q_2 = \{\mathsf{Husband}(x), \mathsf{Woman}(x)\}$. We compute $\rho(R)$ for all $R \in \mathcal{S}_{\mathcal{T}}^{E}$, yielding $Q_3 = \{\mathsf{marries}(x, y_1), \mathsf{marries}(x, y_2), y_1 \neq y_2\}$. Eventually, we obtain an inconsistency-rewritten set of BCQs for an arbitrary atomic concept in $\mathcal{O}$, which is $\mathcal{S} = \{Q_1, Q_2, Q_3\}$.

The expressivity of the IFO-rewritable class is actually higher than that of the first-order rewritable class. In the following, we present two sufficient conditions for detecting IFO-rewritable ontologies that are not first-order rewritable.

A Condition Based on Reachability from Constraints and EGDs. To facilitate finding inconsistency-responsible subsets of a consistent DL ontology $\mathcal{O}$, we restrict $\mathcal{O}$ to be a datalog$^{\pm}$-translatable ontology. In such an ontology, the inconsistencies are caused by constraints or EGDs translated from the TBox and inconsistency-responsible subsets of the TBox can be found by backward traversal from constraints or EGDs. To explain the method for backward traversal from constraints or EGDs, we introduce the notion of *triggering* below.

Definition 2. *Given a set $\mathcal{S}$ of TGDs, we say a predicate P* triggers *another predicate P' in $\mathcal{S}$ if either there is a TGD $R \in \mathcal{S}$ such that P appears in the body of R and P' appears in the head of R, or there exists a predicate P'' such that P triggers P'' and P'' triggers P' in $\mathcal{S}$. Moreover, we say a predicate P* triggers *a constraint or an EGD R in $\mathcal{S}$ if P triggers at least one predicate appearing in the body of R in $\mathcal{S}$; we say a TGD $R \in \mathcal{S}$* triggers *a constraint or an EGD R' in $\mathcal{S}$ if there is a predicate appearing in the head of R triggers R' in $\mathcal{S}$.*

For a constraint or an EGD R and a set $\mathcal{S}$ of TGDs, by $\mathsf{trg}(R, \mathcal{S})$ we denote the set of TGDs in $\mathcal{S}$ that trigger R in $\mathcal{S}$. Then $\mathsf{trg}(R, \mathcal{S})$ is essentially the unique maximal subset of $\mathcal{S}$ that can be backward traversed from R. By considering $\mathsf{trg}(R, \mathcal{S})$ for all constraints or EGDs R rather than the original TBox, we can obtain a condition for detecting IFO-rewritable ontologies that are not first-order rewritable. The following theorem shows this condition and the corresponding method for computing an inconsistency-rewritten set of BCQs, where $\mathsf{pred}(\mathcal{S})$ denotes the set of predicates appearing in a set $\mathcal{S}$ of existential rules.

Theorem 3. *Let $\mathcal{O}$ be a consistent datalog$^{\pm}$-translatable ontology with TBox $\mathcal{T}$, and A an atomic concept in $\mathcal{O}$. If for all $R \in \mathcal{S}_{\mathcal{T}}^{C}$, $A \notin \mathsf{pred}(\mathsf{trg}(R, \mathcal{S}_{\mathcal{T}}^{D}) \cup \{R\})$ or $\mathsf{trg}(R, \mathcal{S}_{\mathcal{T}}^{D})$ is first-order rewritable, and for all $R \in \mathcal{S}_{\mathcal{T}}^{E}$, $A \notin \mathsf{pred}(\mathsf{trg}(R, \mathcal{S}_{\mathcal{T}}^{D}) \cup \{R\})$ or $\{R\}$ is separable from $\mathsf{trg}(R, \mathcal{S}_{\mathcal{T}}^{D})$, then $\bigcup\{\gamma(\rho(R), \mathsf{trg}(R, \mathcal{S}_{\mathcal{T}}^{D})) \mid R \in \mathcal{S}_{\mathcal{T}}^{C}, A \in \mathsf{pred}(\mathsf{trg}(R, \mathcal{S}_{\mathcal{T}}^{D}))\} \cup \{\rho(R) \mid R \in \mathcal{S}_{\mathcal{T}}^{E}, A \in \mathsf{pred}(\mathsf{trg}(R, \mathcal{S}_{\mathcal{T}}^{D}))\}$ is an inconsistency-rewritten set of BCQs for A in $\mathcal{O}$.*

The condition given in the above theorem can be checked regardless of the ABox in time polynomial in the size of the TBox, because computing $\mathsf{trg}(R, \mathcal{S}_{\mathcal{T}}^{D})$ can be done in time polynomial in the size of $\mathcal{S}_{\mathcal{T}}^{D}$ for every constraint or EGD R. The following example shows a datalog$^{\pm}$-translatable ontology that satisfies this condition for some atomic concept but is not first-order rewritable.

Example 3. Let $\mathcal{O}$ be a consistent datalog$^{\pm}$-translatable ontology whose TBox $\mathcal{T}$ consists of the following three axioms.

$$\exists r.A \sqsubseteq A \qquad A \sqcap B \sqsubseteq \bot \qquad B \sqcap C \sqsubseteq \bot$$

Consider computing an inconsistency-rewritten set of BCQs for C in $\mathcal{O}$ by the method in Theorem 3. We first translate $\mathcal{T}$ to the union of a set $\mathcal{S}_{\mathcal{T}}^{D}$ of TGDs, a set $\mathcal{S}_{\mathcal{T}}^{C}$ of constraints and a set $\mathcal{S}_{\mathcal{T}}^{E}$ of EGDs, where $\mathcal{S}_{\mathcal{T}}^{D} = \{\forall x, y\, r(x,y) \wedge A(y) \rightarrow A(x)\}$, $\mathcal{S}_{\mathcal{T}}^{C} = \{R_1 : \forall x\, A(x) \wedge B(x) \rightarrow,\ R_2 : \forall x\, B(x) \wedge C(x) \rightarrow\}$, and $\mathcal{S}_{\mathcal{T}}^{E} = \emptyset$. Since the CQ $A(x)$ cannot be rewritten to a finite disjunction of CQs $Q_D(x)$ such that $\mathsf{ans}(\mathcal{S}_{\mathcal{T}}^{D} \cup \mathcal{A}, A(x)) = \mathsf{ans}(\mathcal{S}, Q_D(x))$ for all ABoxes $\mathcal{A}$, $\mathcal{S}_{\mathcal{T}}^{D}$ is not first-order rewritable, and nor is $\mathcal{O}$. But we can show that $\mathcal{O}$ is IFO-rewritable for C. Since $\mathsf{trg}(R_1, \mathcal{S}_{\mathcal{T}}^{D}) = \mathcal{S}_{\mathcal{T}}^{D}$ and $\mathsf{trg}(R_2, \mathcal{S}_{\mathcal{T}}^{D}) = \emptyset$, we have $C \notin \mathsf{pred}(\mathsf{trg}(R_1, \mathcal{S}_{\mathcal{T}}^{D}) \cup \{R_1\})$ and $C \in \mathsf{pred}(\mathsf{trg}(R_2, \mathcal{S}_{\mathcal{T}}^{D}) \cup \{R_2\})$. Moreover, since $\mathsf{trg}(R_2, \mathcal{S}_{\mathcal{T}}^{D})$ is empty, it is clearly first-order rewritable. Hence an inconsistency-rewritten set of BCQs for C in $\mathcal{O}$ is $\gamma(\rho(R_2), \mathsf{trg}(R_2, \mathcal{S}_{\mathcal{T}}^{D})) = \{\rho(R_2)\}$, where $\rho(R_2) = \{B(x), C(x)\}$.

A Condition Based on Rooted Subgraphs of the ABox Graph. Let $\mathcal{O}$ be a consistent DL ontology with TBox $\mathcal{T}$ and ABox $\mathcal{A}$, and A an atomic concept. Another condition for guaranteeing the existence of an inconsistency-rewritten set of BCQs for A in $\mathcal{O}$ is that, there is a set $\mathcal{S}$ of small subsets of $\mathcal{A}$ such that the number of assertions in any $S \in \mathcal{S}$ is not greater than a given threshold n and every instance a of $\neg A$ in $\mathcal{O}$ is also an instance of $\neg A$ in $\mathcal{T} \cup S$ for some $S \in \mathcal{S}$. The validness of this condition can be shown as follows. Let $\mathcal{S}_A = \{S_a \cup \{A(a)\} \mid \mathcal{O} \models \neg A(a)\}$ where S_a is an element in $\mathcal{S}$ such that $\mathcal{T} \cup S_a \models \neg A(a)$, then $\mathcal{A} \not\models \bigvee \mathcal{S}_A$, and for all individuals a in $\mathcal{O}$, $\mathcal{O} \cup \{A(a)\} \models \bot$ if and only if $\mathcal{A} \cup \{A(a)\} \models \bigvee \mathcal{S}_A$. By $\mathsf{lift}(S)$ we denote the set of atoms obtained from a set S of assertions by replacing different individuals in S with different variables. Let $\mathcal{S}'_A = \{\mathsf{lift}(S) \mid S \in \mathcal{S}_A\}$, then the size of $\mathcal{S}'_A$ is independent of the size of $\mathcal{A}$ and at most polynomial in the size of $\mathcal{T}$ with an exponent not greater than $n+1$. If $\mathcal{T} \cup S \models \bot$ implies $\mathcal{T} \cup \mathsf{lift}(S)\,\theta \models \bot$ for all ground substitutions θ for $\mathsf{lift}(S)$, $\mathcal{S}'_A$ will be an inconsistency-rewritten set of BCQs for A.

To satisfy the above condition as possible, we restrict $\mathcal{O}$ to be an $\mathcal{ELR}_{\bot}$ ontology made up of an $\mathcal{ELR}_{\bot}$ TBox and a normalized ABox. An $\mathcal{ELR}_{\bot}$ TBox consists of role inclusions of the form $r_1 \circ \ldots \circ r_k \sqsubseteq s$ and concept inclusions of the form $C \sqsubseteq D$, where $k \geq 1$, $r_1, \ldots, r_k$, s are atomic roles, and C and D are $\mathcal{EL}_{\bot}$ concepts recursively constructed by $\bot$, $\top$, atomic concepts, existential restrictions $\exists s.C$ and concept conjunctions $C \sqcap D$. The above condition requires computing small ABox subsets preserving instances of $\neg A$ in the ontology. In an $\mathcal{ELR}_{\bot}$ ontology, these ABox subsets can be treated as *maximal rooted subgraphs* of the *ABox graph*. Before showing this result, we formally provide the related notions below.

Definition 3. *The* ABox graph *of a normalized ABox $\mathcal{A}$, denoted by $\mathcal{G}(\mathcal{A})$, is a graph $G = (V, E, L)$ where $V = \mathsf{Ind}(\mathcal{A})$ is a set of vertexes, $E = \{(a, b, r) \mid r(a, b) \in \mathcal{A}\}$ is a set of labeled edges, and $L \colon \mathsf{Ind}(\mathcal{A}) \mapsto 2^{\mathsf{Cn}(\mathcal{A})}$ is a label function such that $L(a) = \{A \mid A(a) \in \mathcal{A}\}$ for all $a \in V$. We say a graph $G' = (V', E', L')$*

is a subgraph *of $G = (V, E, L)$, wirtten $G' \subseteq G$, if $V' \subseteq V$, $E' \subseteq E$ and for all $a \in V'$, $L'(a) \subseteq L(a)$. We say an individual a has a* path *to another individual b in $G = (V, E, L)$, if there is a sequence of labeled edges (a, a_1, r_0), (a_1, a_2, r_1), $\dots$, (a_n, b, r_n) in E. A* root *a of a graph $G = (V, E, L)$ is an individual in V such that for all individuals b other than a in V, a has a path to b in G. A* rooted subgraph *G' of G is a subgraph of G that has at least one root; it is* maximal *if there is no rooted subgraph G'' of G such that $G' \subseteq G''$ and $G'' \not\subseteq G'$.*

Every normalized ABox has a one-to-one mapping to its ABox graph. By $\mathcal{G}^-(G)$ we denote the unique ABox mapped from an ABox graph $G = (V, E, L)$, i.e., $\mathcal{G}^-(G) = \{A(a) \mid a \in V, A \in L(a)\} \cup \{r(a, b) \mid (a, b, v) \in E\}$.

Example 4. Suppose an ABox $\mathcal{A}$ consists of the following 5+m assertions.

$$r(a_1, a_2) \quad r(a_2, a_1) \quad r(a_1, b) \quad r(a_2, b) \quad A(b) \quad r(c_1, b) \quad \dots \quad r(c_m, b)$$

The ABox graph $\mathcal{G}(\mathcal{A})$ of $\mathcal{A}$ is shown below. We can see that $\mathcal{G}(\{r(a_1, a_2)$, $r(a_2, a_1)$, $r(a_1, b)$, $r(a_2, b)$, $A(b)\})$ is a maximal rooted subgraph of $\mathcal{G}(\mathcal{A})$ which has two roots a_1 and a_2; moreover, for all $1 \le i \le m$, $\mathcal{G}(\{r(c_i, b), A(b)\})$ is also a maximal rooted subgraph of $\mathcal{G}(\mathcal{A})$ with the unique root c_i.

$$\begin{tikzcd}
L(a_1) = \emptyset & a_1 \arrow[d, shift right, "r"'] \arrow[dr, "r"] & & c_1 \arrow[dl, "r"] \arrow[d, dash, dotted] & L(c_i) = \emptyset \text{ for all } 1 \le i \le m \\
L(a_2) = \emptyset & a_2 \arrow[u, shift right, "r"'] \arrow[r, "r"] & b & c_m \arrow[l, "r"] & L(b) = \{A\}
\end{tikzcd}$$

In general, all maximal rooted subgraphs of an ABox graph can be retrieved in time polynomial in the size of the ABox, by first identifying all roots of maximal rooted subgraphs and then computing all *full subgraphs* led by these roots. A *full subgraph* of $G = (V, E, L)$ led by an individual a is a subgraph $G' = (V', E', L')$ of G such that $V' = \{a\} \cup \{b \mid a \text{ has a path to } b \text{ in } G\}$, $E' = \{(b, c, r) \in E \mid b \in V', c \in V'\}$ and $L'(b) = L(b)$ for all $b \in V'$. There is a unique full subgraph led by a certain individual. The roots of maximal rooted subgraphs are identified as those individuals a having paths to any individual that has paths to a in the ABox graph. The maximal rooted subgraph in which a is a root can be defined as the full subgraph led by a. The following theorem shows the correctness of this method for computing all maximal rooted subgraphs.

Theorem 4. *Let $G = (V, E, L)$ be an ABox graph and a an individual in V. Then (1) if a is a root of some maximal rooted subgraph of G, then a has paths to any individual that has paths to a in G; (2) if a has paths to any individual that has paths to a in G, then the full subgraph of G led by a is a maximal rooted subgraph of G in which a is a root.*

The following lemma shows that the required set of ABox subsets that preserves all instances of $\neg A$ in an $\mathcal{ELR}_\perp$ ontology can be defined as the set of $\mathcal{G}^-(G)$ for G a maximal rooted subgraph of the ABox graph.

```
Algorithm. ComputeInconsistencyRewrittenSet($\mathcal{T}$, $\mathcal{A}$, $A$)
Input: An $\mathcal{ELR}_\perp$ ontology with TBox $\mathcal{T}$ and ABox $\mathcal{A}$, and an atomic concept $A$.
Output: An inconsistency-rewritten set of BCQs for $A$ in $\mathcal{T} \cup \mathcal{A}$.
1: $\mathcal{S}_Q \leftarrow \emptyset$;
2: for each $G$ in NonIsomorphicMRS($\mathcal{G}(\mathcal{A})$) and each individual $a$ in $G$ do
3:   if $A(a) \notin \mathcal{A}$ and there is no $Q \in \mathcal{S}_Q$ and ground substitution $\theta$ for $Q$ such that
       $Q\theta \subseteq \mathcal{G}^-(G) \cup \{A(a)\}$ then
4:     if $\mathcal{T} \cup \mathcal{G}^-(G) \models \neg A(a)$ then
5:       $Q \leftarrow \mathsf{lift}(\mathcal{G}^-(G) \cup \{A(a)\})$; // replace diff. individuals with diff. variables
6:       for each $Q' \in \mathcal{S}_Q$ such that $Q\theta \subseteq Q'$ for some substitution $\theta$ for $Q$ do
7:         $\mathcal{S}_Q \leftarrow \mathcal{S}_Q \setminus \{Q'\}$;
8:       $\mathcal{S}_Q \leftarrow \mathcal{S}_Q \cup \{Q\}$;
9: return $\mathcal{S}_Q$;
```

Fig. 1. The algorithm for computing an inconsistency-rewritten set of BCQs

Lemma 1. *Given an atomic concept A and a consistent $\mathcal{ELR}_\perp$ ontology $\mathcal{O}$ with TBox $\mathcal{T}$ and ABox $\mathcal{A}$, if a is an instance of $\neg A$ in $\mathcal{O}$, then there is a maximal rooted subgraph G of $\mathcal{G}(\mathcal{A})$ such that $\mathcal{T} \cup \mathcal{G}^-(G) \models \neg A(a)$.*

By the above lemma, we can use an integer threshold n to determine (in PTime in data complexity) if a consistent $\mathcal{ELR}_\perp$ ontology has an inconsistency-rewritten set of BCQs for an arbitrary atomic concept. That is, if the number of assertions in any maximal rooted subgraph of the ABox graph is not greater than n, then we can find an inconsistency-rewritten set of BCQs for a given atomic concept by the algorithm shown in Fig. 1.

The resulting set $\mathcal{S}_Q$ of the algorithm `ComputeInconsistencyRewrittenSet`($\mathcal{T}$, $\mathcal{A}$, A) only keeps the most general BCQs, where a set of atoms (or BCQ) Q is said to be *more general* than another set of atoms (or BCQ) Q' if there is a substitution θ for Q such that $Q\theta \subseteq Q'$. To avoid generating equivalent BCQs up to renaming of variables, the algorithm only handles non-isomorphic subgraphs of $\mathcal{G}(\mathcal{A})$, where two subgraphs G_1 and G_2 are said to be *isomorphic* if $\mathsf{lift}(\mathcal{G}^-(G_1))$ and $\mathsf{lift}(\mathcal{G}^-(G_2))$ are equivalent up to renaming of variables. In the algorithm, `NonIsomorphicMRS`($\mathcal{G}(\mathcal{A})$) denotes the set of non-isomorphic maximal rooted subgraphs of $\mathcal{G}(\mathcal{A})$, i.e., any two subgraphs in this set are not isomorphic. The cardinality of this set is at most polynomial in the size of $\mathcal{T}$ with an exponent not greater than n. The algorithm handles all subgraphs G in `NonIsomorphicMRS`($\mathcal{G}(\mathcal{A})$) and all individuals a in G to construct $\mathcal{S}_Q$. In case $A(a) \in \mathcal{A}$, since $\mathcal{T} \cup \mathcal{A}$ is consistent and $\mathcal{G}^-(G) \cup \{A(a)\} \subseteq \mathcal{A}$, $\mathcal{T} \cup \mathcal{G}^-(G) \cup \{A(a)\}$ is also consistent, i.e., $\mathcal{T} \cup \mathcal{G}^-(G) \not\models \neg A(a)$. In case there is some $Q \in \mathcal{S}_Q$ that is more general than $\mathcal{G}^-(G) \cup \{A(a)\}$, since no individual occurs in Q, the BCQ $\mathsf{lift}(\mathcal{G}^-(G) \cup \{A(a)\})$ is less general than Q and thus is not added to $\mathcal{S}_Q$. In other cases (lines 4–8), if and only if $\mathcal{T} \cup \mathcal{G}^-(G) \models \neg A(a)$, all BCQs that are less general than $\mathsf{lift}(\mathcal{G}^-(G) \cup \{A(a)\})$ are removed from $\mathcal{S}_Q$, making $\mathcal{S}_Q$ contain only the most general BCQs; moreover, $\mathsf{lift}(\mathcal{G}^-(G) \cup \{A(a)\})$ is added to $\mathcal{S}_Q$.

The algorithm first computes `NonIsomorphicMRS`($\mathcal{G}(\mathcal{A})$) regardless of $\mathcal{T}$ in time polynomial in the size of $\mathcal{A}$, and then computes $\mathcal{S}_Q$ from `NonIsomorphic-MRS`($\mathcal{G}(\mathcal{A})$) regardless of $\mathcal{A}$ in time polynomial in the size of $\mathcal{T}$. The correctness follows from Lemma 1 and the following lemma, as shown in Theorem 5.

Lemma 2. *For an $\mathcal{ELR}_\perp$ TBox $\mathcal{T}$ and a set S of assertions such that $\mathcal{T} \cup S$ is inconsistent, $\mathcal{T} \cup \mathsf{lift}(S)\,\theta$ is inconsistent for all ground substitutions θ for $\mathsf{lift}(S)$.*

Theorem 5. *Given an atomic concept A and a consistent $\mathcal{ELR}_\perp$ ontology $\mathcal{O}$ with TBox $\mathcal{T}$ and ABox $\mathcal{A}$, `ComputeInconsistencyRewrittenSet`($\mathcal{T}$, $\mathcal{A}$, A) returns an inconsistency-rewritten set $\mathcal{S}_Q$ of BCQs for A in $\mathcal{O}$, when $|\mathcal{G}^-(G)|$ is not greater than a fixed constant n for all maximal rooted subgraphs G of $\mathcal{G}(\mathcal{A})$.*

Example 5. Consider the ontology $\mathcal{O}$ given by Example 3 where its ABox $\mathcal{A}$ is given by Example 4. $\mathcal{O}$ is a consistent $\mathcal{ELR}_\perp$ ontology, but it is not first-order rewritable. Suppose the threshold n is 5. We show that there is an inconsistency-rewritten set of BCQs for B in $\mathcal{O}$. The maximal rooted subgraphs of $\mathcal{G}(\mathcal{A})$ given in Example 4 are $G_0, G_1, \ldots, G_m$, where $\mathcal{G}^-(G_0) = \{r(a_1,a_2), r(a_2,a_1), r(a_1,b), r(a_2,b), A(b)\}$ and $\mathcal{G}^-(G_i) = \{r(c_i,b), A(b)\}$ for all $1 \le i \le m$. Since $|\mathcal{G}^-(G_i)| \le n$ for all $0 \le i \le m$, we call `ComputeInconsistencyRewrittenSet`($\mathcal{T}$, $\mathcal{A}$, B). Initially, we have `NonIsomorphicMRS`($\mathcal{G}(\mathcal{A})$) $= \{G_0, G_1\}$ and set $\mathcal{S}_Q = \emptyset$. For $\mathcal{G}^-(G_0)$ and a_1, since $B(a_1) \notin \mathcal{A}$, $\mathcal{S}_Q = \emptyset$ and $\mathcal{T} \cup \mathcal{G}^-(G_0) \models \neg B(a_1)$, we add $Q_1 = \{r(x,y), r(y,x), r(x,z), r(y,z), A(z), B(x)\}$ to $\mathcal{S}_Q$. For $\mathcal{G}^-(G_0)$ and a_2, since $Q_1 \cdot \{x/a_2, y/a_1, z/b\} = \mathcal{G}^-(G_0) \cup \{B(a_2)\}$, a_2 is not handled. For $\mathcal{G}^-(G_0)$ and b, since $B(a_1) \notin \mathcal{A}$, $\mathcal{T} \cup \mathcal{G}^-(G_0) \models \neg B(b)$ and Q_1 is not more general than $\mathcal{G}^-(G_0) \cup \{B(b)\}$, we add $Q_2 = \{r(x,y), r(y,x), r(x,z), r(y,z), A(z), B(z)\}$ to $\mathcal{S}_Q$. For $\mathcal{G}^-(G_1)$ and c_1, since $B(c_1) \notin \mathcal{A}$ and $\mathcal{T} \cup \mathcal{G}^-(G_1) \models \neg B(c_1)$, we add $Q_3 = \{r(x,y), A(y), B(x)\}$ to $\mathcal{S}_Q$; moreover, since Q_3 is more general than Q_1, we remove Q_1 from $\mathcal{S}_Q$. For $\mathcal{G}^-(G_1)$ and b, since $B(b) \notin \mathcal{A}$ and $\mathcal{T} \cup \mathcal{G}^-(G_1) \models \neg B(b)$, we add $Q_4 = \{r(x,y), A(y), B(y)\}$ to $\mathcal{S}_Q$; moreover, since Q_4 is more general than Q_2, we remove Q_2 from $\mathcal{S}_Q$. Finally we get $\mathcal{S}_Q = \{Q_3, Q_4\}$, which is an inconsistency-rewritten set of BCQs for B in $\mathcal{O}$.

The DL $\mathcal{ELR}_\perp$ is a core of the EL profile of OWL 2, roughly corresponding to this profile without range restrictions and nominals. However, extending the algorithm in Fig. 1 to deal with range restrictions or nominals is hard or even impossible. On the one hand, applying the algorithm to $\mathcal{ELR}_\perp$ plus range restrictions or inverse roles is incorrect since Lemma 1 does not hold. For example, consider a consistent ontology $\mathcal{O}$ with TBox $\mathcal{T} = \{\exists r.A \sqcap B \sqsubseteq \perp, \exists s^-.\top \sqsubseteq A\}$ and ABox $\mathcal{A} = \{r(a,b), s(c,b)\}$, where $s^-.\top \sqsubseteq A$ says that the range of s is A. It can be seen that a is an instance of $\neg B$ in $\mathcal{O}$, but it is not an instance of $\neg B$ in either $\mathcal{T} \cup \mathcal{G}^-(G_1)$ or $\mathcal{T} \cup \mathcal{G}^-(G_2)$, where $G_1 = \mathcal{G}(\{r(a,b)\})$ and $G_2 = \mathcal{G}(\{s(c,b)\})$ are the two maximal rooted subgraphs of $\mathcal{G}(\mathcal{A})$. To handle range restrictions or inverse roles, we need to consider maximal connected components of $\mathcal{G}(\mathcal{A})$ in which all edges are treated as undirected, but the sizes of these components can easily be greater than a given threshold. On the other hand, applying the algorithm to $\mathcal{ELR}_\perp$ plus nominals is incorrect since Lemma 2 does not hold. For example, consider a TBox $\mathcal{T} = \{\exists r.\{b\} \sqsubseteq \bot\}$ and a set of assertions $S = \{r(a,b)\}$. It can be seen that $\mathcal{T} \cup S$ is inconsistent, but since $\mathsf{lift}(S) = \{r(x,y)\}$, $\mathcal{T} \cup \mathsf{lift}(S)\,\theta = \mathcal{T} \cup \{r(a,a)\}$ is consistent for $\theta = \{x/a, y/a\}$. To extend the algorithm to handle nominals, we need to keep individuals in the resulting set of BCQs. But then it is hard to guarantee that the resulting set is an inconsistency-rewritten set since its size may depend on the size of the ABox.

Handling General Negated Concepts. The proposed method for instance retrieval for atomic negations can be extended to handle general negated concepts $\neg C$. It can be seen that the set of instances of $\neg C$ in a DL ontology $\mathcal{O}$ amounts to the set of instances of $\neg P_C$ in $\mathcal{O} \cup \{P_C \sqsubseteq C\}$, where P_C is a fresh atomic concept not in $\mathcal{O}$. Therefore, the proposed method still works for retrieving instances of $\neg C$ as long as $\mathcal{O} \cup \{P_C \sqsubseteq C\}$ is an IFO-rewritable ontology for P_C.

4 Experimental Evaluation

The goal of our preliminary experiments is to verify if the proposed method is significantly more efficient and scalable than existing methods in retrieving instances of all atomic negations. We focused on first-order rewritable ontologies and implemented the method in JAVA (based on Theorems 1&2), using the query-rewriting system Rapid [7] to compute inconsistency-rewritten sets and using the database system MySQL to store and access ABoxes. We call the implemented system *REwriting-Based System for Instance Retrieval* (*REBSIR*).

We collected two groups of ontologies, where one group was from the Lehigh University Benchmark (LUBM) [11] and the other from DBPedia (version 2014)[1] [3]. Since Rapid cannot handle axioms about concrete roles (i.e. datatype properties), we removed axioms about concrete roles from both the LUBM TBox and the DBPedia TBox, rendering them first-order rewritable. In addition, since there is no constraint translated from the LUBM TBox and none of the atomic negations has instances in LUBM ontologies, we added disjointness axioms to the LUBM TBox for every two sibling atomic concepts in the concept hierarchy such that they have no common instances in any original LUBM ontology. At last we obtained five consistent ontologies named LUBMdn ($n = 1, 5, 10, 50, 100$) for the first group, where n is the number of universities. For the second group, we dumped basic assertions about atomic concepts and abstract roles (i.e. object properties) in the DBPedia TBox from DBPedia-as-Tables[2] to construct the ABox. Since the downloaded DBPedia TBox and DBPedia-as-Tables were generated separately, our constructed ABox was inconsistent with the TBox. To restore consistency, we first computed all minimal conflicts (i.e. minimal subsets of the ABox that are inconsistent with the TBox) by using the rewriting-based method proposed in [8], then removed from the ABox a small hitting set S for the set of minimal conflicts. S was iteratively computed by, in each iteration, adding

[1] http://downloads.dbpedia.org/2014/dbpedia_2014.owl.bz2
[2] http://web.informatik.uni-mannheim.de/DBpediaAsTables/

Table 1. The statistics about test ontologies

Ontology	#C	#R	#TA	#AA	#I
LUBMd1–LUBMd100	43	25	158	100,543–13,824,437	17,174–2,179,766
DBPedia1%–DBPedia100%	811	1,309	3,679	141,039–14,164,192	116,683–3,695,525

Note: #C/#R/#TA/#AA/#I is the number of atomic concepts/abstract roles/ TBox axioms/ABox assertions/individuals.

Table 2. The number of timeout cases (* for running out of memory)

System	LUBMd1	LUBMd5	LUBMd10	DBPedia1%	DBPedia5%	DBPedia10%
FaCT++	0	21	19	0	1	1
KAON2	0	0	0	63	400	734
Pellet	12	12	43	0	0	*811
HermiT	36	38	38	811	811	811

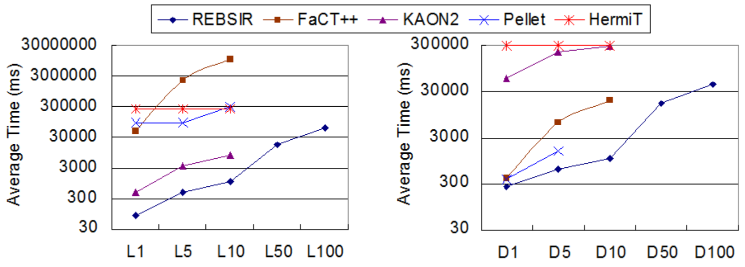
to S an assertion that locates in the most minimal conflicts without any element in S. Finally, we kept $n\%$ of assertions in the modified ABox and obtained five consistent ontologies named DBPedia$n\%$, where $n = 1, 5, 10, 50, 100$. The statistics about all test ontologies are summarized in Table 1.[3]

We first compared the proposed method with the common method implemented in most state-of-the-art DL systems. For every atomic negation $\neg A$, the common method first adds a new axiom $\neg A \sqsubseteq P_A$ to the TBox and then retrieves instances of P_A, where P_A is a fresh atomic concept. Since the performance of the common method may vary with different DL systems, we compared REBSIR with FaCT++ (version 1.6.3) [19], KAON2 (version 2008-06-29) [14], Pellet (version 2.3.1) [18] and HermiT (version 1.3.8) [9]. We set a time limit of five minutes for retrieving instances of a single atomic negation. All experiments were conducted on a laptop with Intel Dual-Core 2.60GHz CPU and 8GB RAM, running Windows 7 with the maximum Java heap size set to 8GB.

For the LUBMd (resp. DBPedia) TBox, REBSIR computes an inconsistency-rewritten set made up of 3,178 (resp. 865,437) BCQs in 200 milliseconds (resp. 65 seconds). This computation is done once before retrieving instances of any atomic negation in a test ontology.

All compared systems except REBSIR run out of memory for LUBMd50, LUBMd100, DBPedia50% and DBPedia100%, and sometimes exceed the time limit of five minutes for other ontologies. Table 2 reports the number of timeout cases. REBSIR has no timeout cases, thus it does not appear in the table. Figure 2 shows the comparison results on the average execution time (in milliseconds) for retrieving instances of an atomic negation. The displayed execution time for REBSIR includes the equally shared time for computing the inconsistency-rewritten set, while the execution time for other systems excludes

[3] All test ontologies and compared systems including REBSIR and others are available at http://www.dataminingcenter.net/rebsir/.



Note: Ln and Dn are respectively short for LUBMdn and DBPedia$n$%. The average execution time is not shown if the system runs out of memory in a test ontology.

Fig. 2. The average execution time for retrieving instances of an atomic negation

the time for loading the test ontology. Moreover, for all compared systems except FaCT++, the execution time in a timeout case is approximated as five minutes since these systems are forced to stop handling the current atomic negation and start handling the next atomic negation once timeout occurs.

As can be seen in Fig. 2, FaCT++ works much better on DBPedia$n$% than on LUBMdn. Pellet also works better on DBPedia$n$% than on LUBMdn, except that for DBPedia10% it runs out of memory. KAON2 works much better on LUBMdn than on DBPedia$n$%, possibly because the resolution-based method used in KAON2 works worse with more axioms in the TBox. HermiT works sightly better on LUBMdn than on DBPedia$n$%. It cannot finish within the time limit for any atomic negation in any DBPedia ontology. Our system REBSIR is the best among all compared systems. It is significantly more efficient than other compared systems and scales to tens of millions of assertions. In particular, it only spends one minute or so on average to retrieve instances of an atomic negation in the two largest ontologies that have more than ten million assertions.

We also compared the proposed method with the consistency checking (CC-) based method under the same test environment. For retrieving instances of an atomic negation $\neg A$ in a consistent ontology $\mathcal{O}$, the CC-based method retrieves the set S of instances of A in $\mathcal{O}$, and then for every individual a occurring in $\mathcal{O}$ but not in S, checks if $\mathcal{O} \cup \{A(a)\} \models \bot$ to determine instances of $\neg A$ in $\mathcal{O}$. We implemented the CC-based method in the state-of-the-art DL system ELK (version 0.4.1) [15], which is highly optimized for DLs in the $\mathcal{EL}$ family [1]. We removed axioms that are not supported by ELK from LUBMdn and kept DBPedia$n$% intact. In this experiment, REBSIR has a similar performance on the modified test ontologies, but ELK (with the default four workers) cannot finish retrieving instances of any atomic negation in any test ontology within the time limit. Although ELK is highly efficient in performing a consistency check, it fails to perform tens of thousands of consistency checks within the time limit and is impractical for instance retrieval for negated concepts.

5 Related Work

In DL systems, instance retrieval for a concept C is often reduced to instance retrieval for an atomic concept P_C by adding $C \sqsubseteq P_C$ to the TBox and then solved by certain reasoning methods. Tableau-based methods [2] are common reasoning methods implemented in state-of-the-art DL systems such as FaCT++ [19], Pellet [18] and HermiT [9]. In [13] existing optimization techniques are summarized and some new optimization techniques are proposed for instance retrieval inside tableau-based methods. In [21] a filter-and-refine paradigm is proposed for optimizing instance retrieval inside tableau-based methods. It first computes obvious non-instances and obvious instances and then performs instance checking for remaining candidate instances. We do not explicitly compare the above optimization techniques with our proposed method because these techniques have at least partially been implemented in most DL systems. For reasoning in expressive DLs, resolution-based methods [14] are another popular paradigm, implemented in a modern DL system KAON2. Our experimental results have shown that our proposed method is more efficient and scalable than both tableau-based and resolution-based methods in retrieving instances of all atomic negations.

Instance retrieval is also related to conjunctive query answering (CQA). There are three efficient approaches to CQA. The first approach is query rewriting, which has been adapted in this work. The second approach is materialization, implemented in scalable DL systems such as WebPIE [20]. It computes an ABox completion containing all ABox consequences wrt the TBox so that subsequent reasoning can be performed in the ABox completion only. The last approach (see e.g. [16]) is a combination of query rewriting and materialization, which first approximates an ABox completion, then rewrites the given query to another one so as to filter out incorrect answers. Except for a query rewriting method proposed in [10], all the above approaches are only applicable to Horn fragments of DLs and cannot be applied to instance retrieval for negated concepts due to the necessity of adding non-Horn features (i.e. concept disjunctions) to the given ontology. The method proposed in [10], however, involves a rather complicated step for transforming disjunctive datalog to datalog and has no evaluation result by now. Recently, a hybrid approach to CQA is proposed in [22]. It first computes a lower bound and an upper bound of the set of answers, then computes answers between the two bounds. Similarly, it needs to add non-Horn features to the ontology before applied to instance retrieval for negated concept, making the complexity beyond PTime in data complexity. In [12] the CQA problem extended by negative atoms is studied for two simple DLs in the DL-Lite family. However, the study [12] focuses on the computational complexity and does not provide practical solutions to the extended CQA problem.

6 Conclusion and Future Work

In this paper we have studied a new approach to instance retrieval for negated concepts based on query rewriting. We identified the class of IFO-rewritable ontologies which guarantees that instance retrieval for an atomic negation can be reduced to answering a disjunction of CQs over the ABox. To show that the IFO-rewritable class is more expressive than the first-order rewritable class, we presented two sufficient conditions for detecting IFO-rewritable ontologies that are not first-order rewritable. An IFO-rewritable ontology $\mathcal{O}$ for an atomic concept A is characterized by an inconsistency-rewritten set of BCQs for A, which witnesses the inconsistency of $\mathcal{O} \cup \{A(a)\}$ for all instances a of $\neg A$ in $\mathcal{O}$. We empirically showed that using inconsistency-rewritten sets makes instance retrieval for all atomic negations more efficient and scalable than existing methods.

For future work, we plan to conduct extensive experiments on more IFO-rewritable ontologies. Moreover, we plan to develop incremental methods for computing inconsistency-rewritten sets. Finally, we intend to discover more sufficient conditions for detecting IFO-rewritable ontologies. In particular, we plan to relax the second sufficient condition from $\mathcal{ELR}_\perp$ to more expressive DLs by considering concept disjunctions and cardinality restrictions.

Acknowledgments. This work is partly supported by NSFC (61375056), Guangdong Natural Science Foundation (S2013010012928) and the EC MSC K-Drive project (286348).

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: IJCAI, pp. 364–369 (2005)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the web of data. J. Web Sem. **7**(3), 154–165 (2009)
4. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. J. Web Sem. **14**, 57–83 (2012)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web **2**(1), 43–53 (2011)
6. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning **39**(3), 385–429 (2007)
7. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting for OWL 2 QL. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 192–206. Springer, Heidelberg (2011)
8. Du, J., Wang, K., Shen, Y.: Towards tractable and practical ABox abduction over inconsistent description logic ontologies. In: AAAI, pp. 1489–1495 (2015)

9. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 reasoner. J. Autom. Reasoning **53**(3), 245–269 (2014)
10. Grau, B.C., Motik, B., Stoilos, G., Horrocks, I.: Computing datalog rewritings beyond horn ontologies. In: IJCAI, pp. 832–838 (2013)
11. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Sem. **3**(2–3), 158–182 (2005)
12. Gutiérrez-Basulto, V., Ibañez-García, Y., Kontchakov, R., Kostylev, E.V.: Conjunctive queries with negation over DL-Lite: a closer look. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 109–122. Springer, Heidelberg (2013)
13. Haarslev, V., Möller, R.: On the scalability of description logic instance retrieval. J. Autom. Reasoning **41**(2), 99–142 (2008)
14. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive datalog. J. Autom. Reasoning **39**(3), 351–384 (2007)
15. Kazakov, Y., Krötzsch, M., Simancik, F.: The incredible ELK - from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. J. Autom. Reasoning **53**(1), 1–61 (2014)
16. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic EL using a relational database system. In: IJCAI, pp. 2070–2075 (2009)
17. Pan, J.Z., Ren, Y., Jekjantuk, N., Garcia, J.: Reasoning the FMA ontologies with TrOWL. In: ORE, pp. 107–113 (2013)
18. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Sem. **5**(2), 51–53 (2007)
19. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
20. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.E.: WebPIE: A web-scale parallel inference engine using mapreduce. J. Web Sem. **10**, 59–75 (2012)
21. Wandelt, S., Möller, R., Wessel, M.: Towards scalable instance retrieval over ontologies. Int. J. Software and Informatics **4**(3), 201–218 (2010)
22. Zhou, Y., Nenov, Y., Grau, B.C., Horrocks, I.: Pay-as-you-go OWL query answering using a triple store. In: AAAI, pp. 1142–1148 (2014)