The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

A Framework and Positive Results for IAR-Answering

Despoina Trivela
Athens University of
Economics and Business
Athens, Greece
despoina@aueb.gr

Giorgos Stoilos
Babylon Health
London, SW3 3DD, UK

Vasilis Vassalos
Athens University of
Economics and Business
Athens, Greece
vassalos@aueb.gr

Abstract

Inconsistency-tolerant semantics, like the IAR semantics, have been proposed as means to compute meaningful query answers over inconsistent Description Logic (DL) ontologies. So far query answering under the IAR semantics (IAR-answering) is known to be tractable only for arguably weak DLs like DL-Lite and the quite restricted $\mathcal{EL}_{\bot nr}$ fragment of $\mathcal{EL}_{\bot}$. Towards providing a systematic study of IAR-answering, in the current paper we first present a general framework/algorithm for IAR-answering which applies to arbitrary DLs but need not terminate. Nevertheless, this framework allows us to develop a sufficient condition for tractability of IAR-answering and hence of termination of our algorithm. We then show that this condition is *always* satisfied by the arguably expressive DL DL-Lite$_{\mathsf{bool}}$, providing the first positive result for IAR-answering over a non-Horn-DL. In addition, recent results show that this condition usually holds for real-world ontologies and techniques and algorithms for checking it in practice have also been studied recently; thus, overall our results are highly relevant in practice. Finally, we have provided a prototype implementation and a preliminary evaluation obtaining encouraging results.

Introduction

Answering queries over data described using Description Logic (DL) ontologies has recently received significant attention. In the vast majority of cases the problem has been studied over consistent datasets (Calvanese et al. 2007; Pérez-Urbina, Motik, and Horrocks 2010; Kikot, Kontchakov, and Zakharyaschev 2012; Trivela et al. 2015). However, in real-world applications datasets may often be inconsistent with respect to the axioms specified in the ontology because, e.g., they may originate from different sources or generated automatically from an information extraction module.

In order to be able to provide "meaningful" answers to user queries even in the presence of inconsistencies the so-called inconsistency-tolerant semantics have been proposed (Arenas, Bertossi, and Chomicki 1999; Bertossi 2006; Lembo et al. 2011; Bienvenu and Rosati 2013). Examples are the IAR, ICAR, and AR semantics (Lembo et al. 2011; 2010), which are based in the notion of *repair*, that is a maximal *consistent subset* of the original dataset. Among them the IAR semantics demonstrate nice computational properties as the problem over DL-Lite ontologies is in AC^0 w.r.t. data complexity, while when using the AR semantics it is already coNP-complete.

Nevertheless, the problem of IAR-answering becomes intractable when considering more expressive DLs. More precisely, Rosati (2011) showed that IAR-answering for almost all well-known DLs from $\mathcal{EL}_{\bot}$ to $\mathcal{SHIQ}$ is at least coNP-hard w.r.t. data complexity (in some cases it is even harder for ICAR- and AR-answering). This is to some extent surprising since query answering over *consistent* datasets is known to be tractable for many DLs in that range like $\mathcal{EL}$ and even Horn-$\mathcal{SHIQ}$. To provide a positive result in terms of tractable data complexity Rosati defined $\mathcal{EL}_{\bot nr}$ which is currently the only positive tractability result for IAR-answering over a DL different than DL-Lite.

In the current paper we study IAR-answering over DL-based ontologies attempting to shed light why the problem is so difficult and identify positive tractable results. First, we provide a general algorithm for computing IAR-answers over any given DL ontology. The algorithm is an extension of the one by Lembo et al. (2015) and obviously need not terminate. However, if it terminates then the output is a first-order structure (a disjunctive datalog program extended with negative body atoms) which if evaluated over the data it computes the IAR-answers. Second, using this algorithm we are able to pinpoint the main reason for the difficulty of IAR-answering and devise a sufficient condition for its termination. Interestingly our condition is related to UCQ-rewritability a notion that has been studied quite extensively (Artale et al. 2009; Bienvenu, Lutz, and Wolter 2013; Hansen et al. 2015). More precisely, we can already show that this condition is always satisfied by ontologies expressed in the DL semi-acyclic-$\mathcal{EL}_{\bot}$ (Bienvenu, Lutz, and Wolter 2012) as well as in DL-Lite$_{\mathsf{bool}}$ (Artale et al. 2009) providing what is, to the best of our knowledge, the first tractability result for IAR-answering in a DL that allows for disjunctions. Third, our condition reveals some deficiencies in the original definition of $\mathcal{EL}_{\bot nr}$ which we redefine. Fourth, even for arbitrary DLs our condition may well be satisfied by a given fixed ontology and recent works provide practical means to check this for a wide range of Horn-DLs (Bienvenu, Lutz, and Wolter 2013; Bienvenu et al. 2014; Hansen et al. 2015). All in all, our

results are arguably of much practical relevance and significance. Finally, we have implemented a prototype system and obtained encouraging preliminary evaluation results.

Preliminaries

We use standard terminology like variables, constants, substitutions, renamings, tuples $\vec{t} = (t_1, \ldots, t_n)$, arity of $\vec{t}$, etc. We will also refer to Horn-clauses written as $H \leftarrow \beta_1 \wedge \ldots \wedge \beta_n$, where H is called its head and the set $\{\beta_1, \ldots, \beta_n\}$ is called its body; if H is the false atom (also denoted by $\bot$) then the clause will be called a *negative clause*.

Description Logic

Description Logics (DLs) (Baader et al. 2002) are the theoretical basis of OWL and constitute a family of (mostly) decidable fragments of First-Order Logic. Next we recapitulate the syntax of some well-known DLs and we use $\mathcal{L}$ to denote an arbitrary DL language.

Let $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{I}$ be countable pairwise disjoint sets of atomic concepts (unary predicates), atomic roles (binary predicates) and individuals (constants), respectively. An $\mathcal{EL}_\bot$-concept is inductively defined by the grammar: $C := \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid \exists R.C$, where $A \in \mathbf{C}, R \in \mathbf{R}$ and $C_{(i)}$ are $\mathcal{EL}_\bot$-concepts. An $\mathcal{EL}_\bot$-TBox $\mathcal{T}$ is a finite set of inclusions of the form $C_1 \sqsubseteq C_2$ with C_1, C_2 $\mathcal{EL}_\bot$-concepts. Inclusions of the form $C_1 \sqcap C_2 \sqsubseteq \bot$ (also written as $C_1 \sqsubseteq \neg C_2$) are called *negative* and the rest *positive*. DL-Lite$_{R,\sqcap}$ (or simply DL-Lite) restricts $\mathcal{EL}_\bot$ by disallowing concepts of the form $\exists R.C$, unless C is the top concept $\top$; for R, S roles, DL-Lite also allows for *inverse roles* of the form S^- and role inclusions of the form $S \sqsubseteq R$ and $S \sqsubseteq \neg R$. Finally, DL-Lite$_{\mathsf{bool}}$ extends DL-Lite with concepts of the form $C_1 \sqcup C_2, \neg C$, and $\exists R.\mathsf{self}$.

An ABox $\mathcal{A}$ is a finite set of *assertions* of the form $A(a)$ or $R(a, b)$ where $a, b \in \mathbf{I}, A \in \mathbf{C}$ and $R \in \mathbf{R}$. $\mathcal{A}$ is *consistent* w.r.t. some TBox $\mathcal{T}$ if there exists a model for $\mathcal{T} \cup \mathcal{A}$; otherwise it is *inconsistent*. A *Knowledge Base (KB)* is a set $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$.

The semantics of DLs can be given by a well-known translation to First-Order Logic (FOL) (Baader et al. 2002). Table 1 presents the translation of $\mathcal{EL}_\bot$ and DL-Lite$_{(\mathsf{bool})}$ axioms to First-Order clauses (inverse roles in DL-Lite axioms are omitted). In the following we assume that the TBox axioms are translated into FOL.

Disjunctive Datalog and Conjunctive Queries

A *disjunctive datalog clause* r is a function-free clause of the form $\forall \vec{x}, \vec{y}(\psi(\vec{x}) \leftarrow \phi(\vec{x}, \vec{y}))$ where $\phi(\vec{x}, \vec{y})$ is a conjunction of atoms called the *body* of the clause and $\psi(\vec{x})$ is a disjunction of atoms called its *head*. We will omit variable quantifiers and write $\psi(\vec{x}) \leftarrow \phi(\vec{x}, \vec{y})$. If ψ contains a single atom then the clause is called *datalog*. A *(disjunctive) datalog program* $\mathcal{P}$ is a finite set of (disjunctive) datalog clauses.

A *conjunctive query* (CQ) $\mathcal{Q}$ is a datalog clause with head predicate Q. The variables occuring in Q are called *answer variables*. A *boolean query* $\mathcal{Q}$ is a CQ with no answer variables. An *instance query* is a CQ of the form $Q(x) \leftarrow A(x)$. A UCQ is a finite set of CQs whose answer variables have the same arity. A tuple of constants $\vec{a}$ is a *certain answer* of $\mathcal{Q}$ over a KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ if the arity of $\vec{a}$ agrees with the arity of Q and $\mathcal{T} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$, where $\mathcal{Q}(\vec{a})$ denotes the boolean query obtained by replacing all answer variables in $\mathcal{Q}$ with $\vec{a}$. We use $\mathsf{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$ to denote all certain answers of $\mathcal{Q}$ w.r.t. $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$.

DL Axiom	Clause
$\mathcal{EL}_\bot$ and DL-Lite	
$B_1 \sqcap B_2 \sqsubseteq A$	$A(x) \leftarrow B_1(x) \wedge B_2(x)$
$A \sqsubseteq \exists R.B$	$R(x, f(x)) \leftarrow A(x), B(f(x)) \leftarrow A(x)$
$\exists R \sqsubseteq A$	$A(x) \leftarrow R(x, y)$
$A \sqcap B \sqsubseteq \bot$	$\bot \leftarrow A(x) \wedge B(x)$
$\mathcal{EL}_\bot$	
$\exists R.B \sqsubseteq A$	$A(x) \leftarrow R(x, y) \wedge B(x)$
DL-Lite	
$P \sqsubseteq R$	$R(x, y) \leftarrow P(x, y)$
$P \sqsubseteq \neg R$	$\bot \leftarrow R(x, y) \wedge P(x, y)$
DL-Lite$_{bool}$	
$A \sqsubseteq B_1 \sqcup B_2$	$B_1(x) \vee B_2(x) \leftarrow A(x)$
$\exists R.\mathsf{self} \sqsubseteq A$	$A(x) \leftarrow R(x, x)$ (dually with $A \sqsubseteq \exists R.\mathsf{self}$)

Table 1: Translation of DL axioms into FOL

Definition 1. Let $\mathcal{T}$ be an $\mathcal{L}$-TBox, $\mathcal{A}$ an ABox consistent w.r.t. $\mathcal{T}$ and $\mathcal{Q}$ a CQ. A *disjunctive datalog-rewriting* (or simply *rewriting*) of $\mathcal{Q}$ w.r.t. $\mathcal{T}$ is a disjunctive datalog program $\mathcal{R}$ such that $\mathcal{T} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$ iff $\mathcal{R} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$, or in case $\mathcal{Q}$ is boolean $\mathcal{T} \cup \mathcal{A} \models \mathcal{Q}$ iff $\mathcal{R} \cup \mathcal{A} \models \mathcal{Q}$. We say that a query $\mathcal{Q}$ is *(disjunctive) datalog-rewritable* w.r.t. $\mathcal{T}$ if there exists a (disjunctive) datalog-rewriting $\mathcal{R}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$; if $\mathcal{R}$ is a UCQ, then $\mathcal{Q}$ is called *UCQ-rewritable* w.r.t. $\mathcal{T}$.

The existence of rewritings as well as practical algorithms for computing them have been extensively studied for a wide variety of DLs. For DL-Lite TBoxes one can always compute a UCQ-rewriting (Calvanese et al. 2007) while for $\mathcal{EL}_\bot$ and Horn-$\mathcal{SHIQ}$ a datalog-rewriting (Eiter et al. 2012). Disjunctive programs are required in the case of highly expressive DLs like $\mathcal{SHIQ}$ (Hustadt, Motik, and Sattler 2007), however, still a datalog-rewriting may exist (Cuenca Grau et al. 2013).

IAR semantics

In order to retrieve meaningful answers even from inconsistent ABoxes the so-called inconsistency-tolerant semantics have been introduced. From those we next recapitulate the IAR semantics (Lembo et al. 2015).

Definition 2. A *repair* of a KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ is any maximal (w.r.t. set inclusion) subset of $\mathcal{A}$ that is consistent w.r.t. $\mathcal{T}$. We use $\mathcal{A}_{\mathsf{ir}}$ to denote the *intersection* of all repairs of $\mathcal{K}$. Let $\mathcal{Q}$ be a CQ and let $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ be an KB. A tuple of constants $\vec{a}$ is called an *IAR-answer* of $\mathcal{Q}$ over $\mathcal{K}$ if $\vec{a} \in \mathsf{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}_{\mathsf{ir}})$. We use $\mathsf{cert}_{\mathsf{ir}}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$ to denote the set of all IAR-answers of $\mathcal{Q}$ over $\mathcal{K}$ and we also write $\mathcal{T} \cup \mathcal{A} \models_{\mathsf{ir}} \mathcal{Q}(\vec{a})$.

A Framework for IAR-answering

A straightforward approach to compute the IAR-answers would be to compute $\mathcal{A}_{\mathsf{ir}}$, however, if $\mathcal{A}$ is large or inaccessible (e.g., due to access restrictions) this may be impos-

sible. A different approach proposed in (Lembo et al. 2011; 2015) is to rewrite the input query in such a way that its evaluation over $\mathcal{A}$ would only return the IAR-answers.

Example 1. Let the TBox $\mathcal{T} = \{\bot \leftarrow A(x) \wedge C(x), A(x) \leftarrow P(x,y)\}$, the ABox $\mathcal{A} = \{C(a), P(a,c), A(b)\}$ and the CQ $\mathcal{Q} = Q(x) \leftarrow A(x)$. Clearly, $\mathcal{A}$ is inconsistent w.r.t. $\mathcal{T}$ and the repairs are $\mathcal{A}_{r_1} = \{C(a), A(b)\}$ and $\mathcal{A}_{r_2} = \{P(a,c), A(b)\}$. Hence, we have $\mathcal{A}_{\mathsf{ir}} = \{A(b)\}$ and $\mathsf{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}_{\mathsf{ir}}) = \{b\} = \mathsf{cert}_{\mathsf{ir}}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$. Instead, consider the rewriting $\mathcal{R} = \{Q(x) \leftarrow A(x), Q(x) \leftarrow P(x,y)\}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$. We can note that $\mathsf{cert}(\mathcal{R}, \mathcal{A}) = \{a, b\}$ which, on the one hand, contains the IAR-answer b, however, on the other hand, it contains the non-IAR-answer a. Lembo et al. (Lembo et al. 2011; 2015) noticed that we can exclude such "incorrect" answers by extending the clauses in $\mathcal{R}$ with negative atoms which will prevent the clauses of $\mathcal{R}$ to bind to certain patterns of $\mathcal{A}$. For example, due to $\mathcal{T} \models \bot \leftarrow A(x) \wedge C(x)$ and $\mathcal{T} \models \bot \leftarrow P(x,y) \wedge C(x)$, $\mathcal{R}$ should be extended to $\mathcal{R}^{\neg} = \{Q(x) \leftarrow A(x) \wedge \neg C(x), Q(x) \leftarrow P(x,y) \wedge \neg C(x)\}$. Then, $\mathsf{cert}(\mathcal{R}^{\neg}, \mathcal{A}) = \{b\}$ as required. ♢

The following definition formalises the notion of an IAR-rewriting for a query $\mathcal{Q}$ w.r.t. some TBox $\mathcal{T}$, a structure which when evaluated over a possibly inconsistent ABox returns only the IAR-answers of $\mathcal{Q}$.

Definition 3. Given an $\mathcal{L}$-TBox and a CQ $\mathcal{Q}$, an *IAR-rewriting* $\mathcal{R}^{\mathsf{ir}}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$ is a disjunctive datalog program, possibly extended with negative body atoms, such that for every ABox $\mathcal{A}$ we have $\mathcal{T} \cup \mathcal{A} \models_{\mathsf{ir}} \mathcal{Q}(\vec{a})$ iff $\mathcal{R}^{\mathsf{ir}} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$.

Example 1 suggests that to compute an IAR-rewriting we should at least compute all possible negative clauses $\mathcal{C}$ such that $\mathcal{T} \models \mathcal{C}$ and then use them in order to annotate the clauses of a rewriting with proper negative atoms. We call such an operation a negative closure of $\mathcal{T}$.

Definition 4. A *negative closure* of a $\mathcal{L}$-TBox $\mathcal{T}$, denoted by $\mathcal{T}_{cn}$, is a finite set of negative clauses such that $\mathcal{T} \models \bot \leftarrow \bigwedge \beta_i$ iff some $\bot \leftarrow \bigwedge \alpha_i$ in $\mathcal{T}_{cn}$ exists with $\bot \leftarrow \bigwedge \alpha_i \models \bot \leftarrow \bigwedge \beta_i$.

However, there are two important technical details to be considered when extending the elements of a rewriting with negative atoms (Lembo et al. 2011; 2015). First, one should only take into account negative clauses that have minimal bodies w.r.t. set inclusion. More precisely, a negative clause $\mathcal{C}$ should be considered, only if no other clause is a syntactical subset of $\mathcal{C}$. Second, one should be careful about possible unifications of binary atoms in relation to negative clauses. These two issues are illustrated in the following examples.

Example 2. Let the query $\mathcal{Q} = Q(x) \leftarrow A(x)$, the TBox $\mathcal{T}_1 = \{\mathcal{C}\}$, where $\mathcal{C} = \bot \leftarrow A(x) \wedge B(x) \wedge C(x)$ and the $\mathcal{T}_1$-inconsistent ABox $\mathcal{A} = \{A(a), B(a), C(a)\}$. There are three repairs, $\{A(a), B(a)\}$, $\{A(a), C(a)\}$, and $\{B(a), C(a)\}$, hence $\mathcal{A}_{\mathsf{ir}} = \emptyset$ and $\mathsf{cert}_{\mathsf{ir}}(\mathcal{Q}, \mathcal{T}_1 \cup \mathcal{A}) = \emptyset$.

Following the technique in (Lembo et al. 2011) consider the rewriting $\mathcal{R} = \{Q(x) \leftarrow A(x)\}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}_1$. Due to $\mathcal{C} \in \mathcal{T}_1$, $\mathcal{R}$ should be extended to $\mathcal{R}^{\neg} = \{Q(x) \leftarrow A(x) \wedge \neg(B(x) \wedge C(x))\}$ for which we have $\mathsf{cert}(\mathcal{R}^{\neg}, \mathcal{A}) = \emptyset$ as required.

Algorithm 1 IAR-Rewriting

Input: a CQ $\mathcal{Q}$ and an $\mathcal{L}$-TBox $\mathcal{T}$
Compute a negative closure $\mathcal{T}_{cn}$ of $\mathcal{T}$
$\mathcal{T}'_{cn} := \mathsf{minimise}(\mathsf{saturate}(\mathcal{T}_{cn}))$
Compute a rewriting $\mathcal{R}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$
$\mathcal{R}' := \mathsf{saturate}(\mathcal{R})$
$\mathcal{R}^{\mathsf{ir}} := \emptyset$
for $\mathcal{Q} \in \mathcal{R}'$ **do**
  $\mathcal{Q}^{\neg} := \mathcal{Q}$
  for $\alpha \in \mathcal{Q}$ where α is not an inequality atom **do**
    for $\bot \leftarrow \beta_1 \wedge \ldots \wedge \beta_m \in \mathcal{T}'_{cn}$ **do**
      if $\alpha = \beta_k \mu$, μ a renaming, k in $[1, m]$ **then**
        Add $\neg(\beta_1 \wedge \ldots \wedge \beta_m)\mu$ to $\mathcal{Q}^{\neg}$
      end if
    end for
  end for
  $\mathcal{R}^{\mathsf{ir}} := \mathcal{R}^{\mathsf{ir}} \cup \{\mathcal{Q}^{\neg}\}$
end for
return $\mathcal{R}^{\mathsf{ir}}$

Now consider the TBox $\mathcal{T}_2 = \mathcal{T}_1 \cup \{A(x) \leftarrow B(x)\}$. Interestingly, $\{B(a), C(a)\}$ is no longer a repair of $\mathcal{A}$ and hence $\mathcal{A}'_{\mathsf{ir}} = \{A(a)\}$. Therefore, $\mathsf{cert}_{\mathsf{ir}}(\mathcal{Q}, \mathcal{T}_2 \cup \mathcal{A}) = \{a\}$ and to construct a correct IAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}_2$ one should not add $\neg(B(x) \wedge C(x))$ to $\mathcal{Q}(x) \leftarrow A(x) \in \mathcal{R}$. Notice that in the new TBox we have $\mathcal{T}_2 \models \mathcal{C}'$, where $\mathcal{C}' = \bot \leftarrow B(x) \wedge C(x)$ and, moreover, $\mathcal{C}' \models \mathcal{C}$ and $\mathcal{C}'$ is a syntactical subset of $\mathcal{C}$. ♢

Example 3. Let $\mathcal{T}$ be an $\mathcal{L}$-TBox, a query $\mathcal{Q} = Q(x) \leftarrow R(x,y)$, and the rewriting $\mathcal{R} = \{\mathcal{Q}\}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$. Assume that $\mathcal{T}$ entails the negative clause $\bot \leftarrow R(x,x) \wedge S(x,x)$. Intuitively, one should add to $\mathcal{Q}$ the body atom $\neg S(x,x)$ but only when $x = y$. The set $\mathcal{R}^{\neg}$ consisting of the following queries is an IAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$:

$$\mathcal{Q}_1 = Q(x) \leftarrow R(x,y) \wedge x \neq y$$
$$\mathcal{Q}_2 = Q(x) \leftarrow R(x,x) \wedge \neg S(x,x)$$

Consider the ABoxes $\mathcal{A}_1 = \{R(a,b), S(a,b)\}$, $\mathcal{A}_2 = \{R(a,a), S(a,a)\}$. As expected, $\mathsf{cert}(\mathcal{R}^{\neg}, \mathcal{A}_1) = \{a\}$ and $\mathsf{cert}(\mathcal{R}^{\neg}, \mathcal{A}_2) = \emptyset$. Note that we cannot drop the conjunct $x \neq y$ from $\mathcal{Q}_1$ as then we would incorrectly have $\mathsf{cert}(\mathcal{R}^{\neg}, \mathcal{A}_2) = \{a\}$. ♢

Using the notions of a rewriting and of a negative closure of a TBox our approach for computing an IAR-rewriting is depicted in Algorithm 1. For TBoxes expressed in arbitrary DLs a negative closure may obviously not exist, however, in the next section we will study conditions that ensure its existence.

The algorithm also uses two procedures. Procedure saturate is defined in (Lembo et al. 2015, Algorithm 3) and is related to the issue of distinct variables illustrated in Example 3. Roughly speaking given a clause $\mathcal{C}$, this procedure replaces it with a list of clauses $\mathcal{C}_1, \ldots, \mathcal{C}_n$ such that $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \vee \ldots \vee \mathcal{C}_n$ and for every $\mathcal{C}_i$ if x, y are pairs of distinct variables then $\mathcal{C}_i$ contains the conjunct $x \neq y$. This list of clauses is generated by using unification on the variables of $\mathcal{C}$, hence n can be exponential in the number of variables in $\mathcal{C}$. Finally, minimise is defined as follows:

Definition 5. Let $\mathcal{T}$ be a set of negative clauses. Procedure $\mathsf{minimise}(\mathcal{T})$ returns a new set of clauses $\mathcal{T}'$ such that $\mathcal{T}' \models \mathcal{T}$ and for every $\mathcal{C} \in \mathcal{T}'$ no $\mathcal{C}'$ in $\mathcal{T}'$ different to $\mathcal{C}$ is (up to variable renaming) a syntactical subset of $\mathcal{C}$.

Note that it is important that the minimise procedure is applied after saturate (Lembo et al. 2015) as the latter may introduce non-minimal clauses.

Example 4 illustrates the steps of Algorithm 1.

Example 4. Let the $\mathcal{EL}$-TBox $\mathcal{T}$

$$\bot \leftarrow D(x) \wedge A(x) \quad (1)$$
$$A(x) \leftarrow R(x,y) \wedge B(y) \quad (2)$$

and a query $\mathcal{Q} = Q(x) \leftarrow A(x)$. At first step, Algorithm 1 constructs the datalog-rewriting $\mathcal{R}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$:

$$\begin{aligned}\mathcal{R} \quad = \quad & \{Q(x) \leftarrow A(x), \\ & A(x) \leftarrow R(x,y) \wedge B(y)\}\end{aligned}$$

By resolving clause (2) on (1) Algorithm 1 constructs a negative closure of $\mathcal{T}$ that is, $\mathcal{T}_{cn} = \{\bot \leftarrow D(x) \wedge A(x), \bot \leftarrow D(x) \wedge R(x,y) \wedge B(y)\}$. Next, it applies saturate on $\mathcal{R}$ and $\mathcal{T}_{cn}$, and then minimise on $\mathcal{T}_{cn}$ and constructs the IAR-rewriting:

$$\begin{aligned}\mathcal{R}^{\mathsf{ir}} \quad = \quad & \{Q(x) \leftarrow A(x) \wedge \neg(A(x) \wedge D(x)), \\ & A(x) \leftarrow R(x,y) \wedge x \neq y \wedge B(y) \wedge \neg(R(x,y) \wedge \\ & x \neq y \wedge B(y) \wedge D(x)), \\ & A(x) \leftarrow R(x,x) \wedge B(x) \wedge \neg(R(x,x) \wedge B(x) \\ & \wedge D(x))\}\end{aligned}$$

♢

Theorem 6. *Given an input CQ $\mathcal{Q}$ and $\mathcal{L}$-TBox $\mathcal{T}$, if $\mathcal{Q}$ is disjunctive datalog-rewritable w.r.t. $\mathcal{T}$ and there exists a negative closure $\mathcal{T}_{cn}$ of $\mathcal{T}$, then Algorithm 1 terminates and returns an IAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$.*

Proof. (sketch) If there exists a rewriting $\mathcal{R}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$, and a negative closure $\mathcal{T}_{cn}$, then Algorithm 1 terminates. The output $\mathcal{R}^{\mathsf{ir}}$ is a disjunctive datalog program possibly extended with negative atoms according to lines 5–16. To prove correctness of Algorithm 1 we show that $\mathcal{R} \cup \mathcal{A} \models_{\mathsf{ir}} \mathcal{Q}(\vec{a})$ iff $\mathcal{R}^{\mathsf{ir}} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$; this is done using induction on the evaluation of $\mathcal{R}^{\mathsf{ir}}$ ($\mathcal{R}$) over $\mathcal{A}$. Moreover, the proof also uses results parts of proofs from (Lembo et al. 2015). □

Positive Results for IAR-rewritability

As can be seen from the previous section the main cause of failure of Algorithm 1 is non-existence of a negative closure. This is already the case for rather simple TBoxes expressed in arguably simple DLs.

Example 5. Let $\mathcal{Q} = Q(x) \leftarrow B(x)$ and let also the following $\mathcal{EL}_{\bot}$-TBox $\mathcal{T}$:

$$\bot \leftarrow A(x) \wedge B(x) \quad (3)$$
$$A(x) \leftarrow R(x,y) \wedge A(y) \quad (4)$$

The program consisting of clauses $\mathcal{Q}$, (3) and (4) is a datalog-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$. Assume we attempt to compute $\mathcal{T}_{cn}$ by using resolution. First, we resolve (3) with (4) to obtain $\bot \leftarrow R(x,y) \wedge A(y) \wedge B(x)$; this clause can then be resolved with (4) to derive the clause $\bot \leftarrow R(x,y) \wedge R(y,z) \wedge A(z) \wedge B(x)$. None of the resolvents entails the other, hence $\mathcal{T}_{cn}$ must contain both. Clearly we can create an infinite number of clauses of all of which must belong to $\mathcal{T}_{cn}$. ♢

Intuitively, the main reason for non-existence of $\mathcal{T}_{cn}$ in the above example is the presence of the recursive clause (4). Although, such clauses are not problematic in query answering over consistent ABoxes, in IAR-answering they cause a blow-up in data complexity from P to coNP (Rosati 2011). Recursion is also known to be the critical factor for non-UCQ-rewritability in DL. Indeed, next we show that UCQ-rewritability of the ABox-consistency checking problem implies the existence of a negative closure.

Definition 7. Let an $\mathcal{L}$-TBox $\mathcal{T}$. We say that *ABox-inconsistency is UCQ-rewritable relative to $\mathcal{T}$* if a union of boolean queries $\mathcal{R}$ with head atom Q exists s.t. for every ABox $\mathcal{A}$, $\mathcal{A}$ is inconsistent w.r.t. $\mathcal{T}$ iff $\mathcal{A} \cup \mathcal{R} \models Q$. $\mathcal{R}$ is a *UCQ-rewriting of ABox-inconsistency relative to $\mathcal{T}$*.

Lemma 8. *ABox-inconsistency is UCQ-rewritable relative to an $\mathcal{L}$-TBox $\mathcal{T}$ iff there exists a negative closure $\mathcal{T}_{cn}$ of $\mathcal{T}$.*

Proof. If ABox-inconsistency is UCQ-rewritable relative to $\mathcal{T}$ then let $\mathcal{R}$ be the UCQ-rewriting of ABox-inconsistency relative to $\mathcal{T}$. We can show that a negative closure of $\mathcal{T}$ can be constructed from $\mathcal{R}$ just by replacing the head atoms of clauses in $\mathcal{R}$ with $\bot$.

Let $\mathcal{T}_{cn}$ be constructed from $\mathcal{R}$ as described above and let $\mathcal{T} \models \mathcal{C}$ for some clause $\mathcal{C} = \bot \leftarrow \bigwedge_1^n \beta_i$. For σ an injective instantiation of the variables of $\mathcal{C}$ we have that $\mathcal{T} \cup \mathcal{C}\sigma \models \bot$ or $\mathcal{T} \cup \bigwedge_1^n \beta_i\sigma \models \bot$ i.e., $\{\beta_1\sigma, \ldots, \beta_n\sigma\}$ is inconsistent. Then, some CQ $\mathcal{Q} = Q \leftarrow \bigwedge_1^m \alpha_i$ must exist such that $\{\beta_1\sigma, \ldots, \beta_n\sigma\} \cup \{\mathcal{Q}\} \models Q$. Since Q does not appear anywhere in $\mathcal{T}$ this implies that some mapping μ from the variables of $\mathcal{Q}$ to individuals in $\{\beta_1\sigma, \ldots, \beta_n\sigma\}$ exists such that we have $\{\alpha_1\mu, \ldots, \alpha_m\mu\} \subseteq \{\beta_1\sigma, \ldots, \beta_n\sigma\}$. Since σ is injective we can compute its inverse σ^-; then we have $\{\alpha_1\mu\sigma^-, \ldots, \alpha_m\mu\sigma^-\} \subseteq \{\beta_1\sigma\sigma^-, \ldots, \beta_n\sigma\sigma^-\}$ or $\{\alpha_1\mu\sigma^-, \ldots, \alpha_m\mu\sigma^-\} \subseteq \{\beta_1, \ldots, \beta_n\}$. Consequently, some $\lambda = \mu\sigma^-$ exists such that $\{\alpha_1\lambda, \ldots \alpha_m\lambda\} \subseteq \{\beta_1, \ldots, \beta_n\}$. By construction, $\mathcal{T}_{cn}$ contains a clause of the form $\bot \leftarrow \bigwedge_1^m \alpha_i$ which by the above we have shown that it subsumes $\mathcal{C}$. Moreover, $\mathcal{T}_{cn}$ is finite since $\mathcal{R}$ is finite.

For the opposite direction, from a negative closure $\mathcal{T}_{cn}$ we can construct a UCQ-rewriting for ABox-inconsistency relative to $\mathcal{T}$ by replacing the head atoms $\bot$ of clauses in $\mathcal{T}_{cn}$ with Q. □

The following Lemma provides a characterisation of UCQ-rewritability of the ABox-consistency problem in terms of UCQ-rewritability of query answering over *consistent* ABoxes for the case of Horn-DLs.

Lemma 9. *Let $\mathcal{T}$ be an $\mathcal{L}$-TBox where $\mathcal{L}$ is a Horn-DL. Let the set of $\mathcal{L}$-concepts $\mathcal{S} = \{A_i(x) \mid \bot \leftarrow A_1(x) \wedge \ldots \wedge$*

$A_m(x) \in \mathcal{T}\}$. *If every instance query* $Q(x) \leftarrow A_i(x)$ *in* $\mathcal{S}$ *is UCQ-rewritable w.r.t.* $\mathcal{T}$ *and* consistent *ABoxes, then ABox-inconsistency is UCQ-rewritable relative to* $\mathcal{T}$.

Proof. By Lemma 8 it suffices to show that there exists a negative closure $\mathcal{T}_{cn}$ of $\mathcal{T}$. Clearly, $\mathcal{T}_{cn}$ can be computed by applying resolution with factoring to $\mathcal{T}$. Since $\mathcal{T}$ is expressed in a Horn-DL the following properties hold for such resolution derivations:

- two negative clauses never resolve with each other.
- by resolving a negative with a positive clause the resolvent is always a negative clause.

The above imply that a $\mathcal{T}_{cn}$ can be computed as follows: initialise $\mathcal{T}_{cn}$ to contain all negative clauses $\bot \leftarrow A_1(x) \wedge \ldots \wedge A_m(x) \in \mathcal{T}$ and compute a UCQ-rewriting $\mathcal{R}_i$ for each $Q(x) \leftarrow A_i(x)$ (by assumption this rewriting exists). Finally add to $\mathcal{T}_{cn}$ the clause $\bot \leftarrow \mathcal{R}_1(x) \wedge \ldots \wedge \mathcal{R}_m(x)$. □

Interestingly, in the case of the non-Horn DL-Lite$_{\mathsf{bool}}$ instance query answering is always UCQ-rewritable (Artale et al. 2009). Moreover, Cuenca Grau et al. (2013) designed a goal-oriented procedure that computes a datalog rewriting of a given DL-Lite$_{\mathsf{bool}}$-TBox. Therefore, these two results together with Lemma 9 can be used to show the first ever positive result on IAR-rewritability for a non-Horn DL.

Theorem 10. *Let* $\mathcal{T}$ *be a DL-Lite*$_{\mathsf{bool}}$*-TBox and let* $\mathcal{Q}$ *be an instance query. Then, on input* $\mathcal{T}, \mathcal{Q}$ *Algorithm 1 terminates and computes an IAR-rewriting of* $\mathcal{Q}$ *w.r.t.* $\mathcal{T}$ *that is a datalog program.*

Proof. (sketch) Let $\mathcal{T}$ be an arbitrary DL-Lite$_{bool}$-TBox. By applying the procedure of Cuenca Grau et al. (2013) $\mathcal{T}$ can be transformed into a datalog program whose body is tree-shaped (the latter follows by restricting Theorem 8 and Lemma 20 from (Cuenca Grau et al. 2013) to the particular form of DL-Lite$_{bool}$ clauses we consider here). Moreover, by the results in (Artale et al. 2009) every instance query formed using symbols of $\mathcal{T}$ is UCQ-rewritable hence Lemma 9 can be applied. □

Example 6. Let the DL-Lite$_{\mathsf{bool}}$-TBox $\mathcal{T} = \{\bot \leftarrow R(x,y) \wedge A(x), A(x) \vee D(x) \leftarrow C(x)\}$ and the instance query $Q(x) \leftarrow D(x)$. Using the procedure described in (Cuenca Grau et al. 2013) we can obtain the equisatisfiable datalog program $\mathcal{R} = \{Q(x) \leftarrow D(x), \bot \leftarrow R(x,y) \wedge A(x), D(x) \leftarrow R(x,y) \wedge C(x)\}$. Given $\mathcal{R}$ as an input Algorithm 1 constructs:

$$\begin{aligned}\mathcal{R}^{\mathsf{ir}} \quad = \quad & \{Q(x) \leftarrow D(x),\\ & D(x) \leftarrow R(x,x) \wedge C(x) \wedge \neg(R(x,x) \wedge A(x)),\\ & D(x) \leftarrow R(x,y) \wedge x \neq y \wedge C(x) \wedge\\ & \neg(R(x,y) \wedge x \neq y \wedge A(x))\}\end{aligned}$$

♢

Bienvenu, Lutz, and Wolter (2012) showed that instance queries over so called semi-acyclic-$\mathcal{EL}$-TBoxes are always UCQ-rewritable. Moreover, $\mathcal{EL}$ is a Horn-DL. Therefore, we can use Lemma 9 to show the following.

Theorem 11. *Let* $\mathcal{T}$ *be a semi-acyclic-*$\mathcal{EL}_\bot$*-TBox and let* $\mathcal{Q}$ *be a CQ. Then, on input* $\mathcal{T}$ *and* $\mathcal{Q}$*, Algorithm 1 terminates and computes an IAR-rewriting of* $\mathcal{Q}$ *w.r.t.* $\mathcal{T}$ *that is a datalog program.*

Restricting $\mathcal{EL}_\bot$ to obtain a fragment for which IAR-answering is tractable (w.r.t. data complexity) was also studied by Rosati (2011) who defined $\mathcal{EL}_{\bot nr}$. Its definition follows the same intuitions as above, that is, that no recursions are involved with concepts that appear in negative clauses. The original definition is arguably sketchy and suffers from some technical glitches, hence we re-define $\mathcal{EL}_{\bot nr}$ using our framework. For a better comparison with the original definition and conciseness in the following we use DL notation.

Definition 12. An $\mathcal{EL}_{\bot nr}$-TBox is an $\mathcal{EL}_\bot$-TBox $\mathcal{T}$ such that for every negative clause $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq \bot$ entailed by $\mathcal{T}$, if $C \sqsubseteq A_i$ is also entailed by $\mathcal{T}$ and C contains an occurence of A_i nested into an existentially quantified concept expression, then some $C' \sqsubseteq A_i$ is entailed by $\mathcal{T}$ where C' does not mention A_i and a substitution σ exists such that each concept and role in $C'\sigma$ occurs in C.

Intuitively, if such σ exists then the recursion induced by $C \sqsubseteq A_i$ is superfluous. Compared to Rosati (2011) our definition differs in this last condition, where Rosati required that $C' \sqsubseteq C$.

Example 7. Consider the TBox $\mathcal{T}$ of Example 5. Clearly, it is not in $\mathcal{EL}_{\bot nr}$, but if we extend it with an axiom $\exists R.\top \sqsubseteq A$, then the resulting TBox $\mathcal{T}' = \mathcal{T} \cup \{\exists R.\top \sqsubseteq A\}$ is in $\mathcal{EL}_{\bot nr}$ and it is easy to verify that there exists a negative closure of $\mathcal{T}'$.

In contrast, if we extend $\mathcal{T}$ with the axiom $D \sqsubseteq \exists R.A$, then the obtained TBox $\mathcal{T}'' = \mathcal{T} \cup \{D \sqsubseteq \exists R.A\}$ is not in $\mathcal{EL}_{\bot nr}$ and a negative closure of $\mathcal{T}''$ does not exist for the same reasons illustrated in Example 5. However, according to the definition given in (Rosati 2011) the TBox $\mathcal{T}''$ is in $\mathcal{EL}_{\bot nr}$. ♢

Theorem 13. *Let* $\mathcal{T}$ *be a* $\mathcal{EL}_{\bot nr}$*-TBox and let* $\mathcal{Q}$ *be a CQ. Then, on input* $\mathcal{T}$ *and* $\mathcal{Q}$ *Algorithm 1 terminates and computes an IAR-rewriting of* $\mathcal{Q}$ *w.r.t.* $\mathcal{T}$ *that is a datalog program.*

For arbitrary Horn-DLs that are not always UCQ-rewritable (like, e.g., $\mathcal{EL}$) in order to check the conditions in Lemma 9 we can exploit many recent results in UCQ-rewritability of instance queries over Horn-DLs (Bienvenu, Lutz, and Wolter 2013; Hansen et al. 2015). More precisely, Bienvenu, Lutz, and Wolter (2013) study UCQ-rewritability of a given instance query over a wide range of Horn-DLs, like $\mathcal{EL}_\bot$, $\mathcal{ELI}_\bot$ and Horn-$\mathcal{SHIF}$ and present a preliminary algorithm based on automata. Subsequently, these results were used to design a practical algorithm and conduct an experimental evaluation which showed that for a large number of real-world TBoxes the vast majority of instance queries are UCQ-rewritable (Hansen et al. 2015). Since all the above DLs are Horn one can use systems like Clipper (Eiter et al. 2012) or Rapid (Trivela et al. 2015) to compute a datalog-rewriting for the input TBox, then the Grind

system (Hansen et al. 2015) to check UCQ-rewritability of all relevant instance queries defined in Lemma 9 and, finally, Algorithm 1 to compute an IAR-rewriting.

Finally, we remark about linear-acyclic-$\mathcal{ELU}$ (Kaminski and Grau 2013) a fragment of $\mathcal{EL}$ with disjunctions for which all instance queries of the form $Q(x) \leftarrow A(x)$ are UCQ-rewritable. Unfortunately, linearity breaks if we extend this DL with negative clauses in an effort to define a fragment of $\mathcal{ELU}_\bot$ for which ABox-consistency is UCQ-rewritable. The authors leave open the problem whether acyclicity alone (without linearity) is enough to guarantee UCQ-rewritability but argue that this could be possible. If this is the case then it will not be hard to show that acyclic-$\mathcal{ELU}_\bot$ is IAR-rewritable.

Evaluation

Based on Algorithm 1 we created a prototype system. It is using Rapid (Trivela et al. 2015) to compute a rewriting $\mathcal{R}$ for $\mathcal{Q}$ and $\mathcal{T}$ (line 3 of Algorithm 1) and Grind (Hansen et al. 2015) along with the approach described in Lemma 9 to decide whether it can compute a negative closure $\mathcal{T}_{cn}$. If the negative closure can be computed, then our system proceeds in extending $\mathcal{R}$ with negative conjuncts as described in lines 6-14, otherwise it reports that a negative closure could not be computed. The whole system currently supports $\mathcal{ELH}_\bot^{dr}$ ontologies as this is the language supported by the current implementation of Grind.

Our test ontologies consist of the seven ontologies used in (Hansen et al. 2015). From them envo, FBbi, and SO include negative clauses (axioms) while for the rest (mohse, nbo, Not-Galen, XP) we had to manually add some; we tried to use concepts that appear in various "levels" of the hierarchy of the ontology so that these affect large or small parts of it. Furthermore, we have used $\mathcal{ELH}_\bot^{dr}$ fragments of the ontologies CARO,[1] BFO[2] and Dolce-Lite.[3] Moreover, for each ontology we manually constructed five test queries. Each one of them contains at least one body atom that uses a predicate (concept or role) involved in a negative clause. More precisely, for axioms of the form $B \sqsubseteq \neg C$ we have constructed queries $Q(x) \leftarrow A(x)$ and $Q(x) \leftarrow D(x)$ such that $\mathcal{T} \models A \sqsubseteq B$ and $\mathcal{T} \models B \sqsubseteq D$. We also tried to use concepts that appear low or high in the ontology hierarchy.

Our tool managed to compute a negative closure for all ontologies except SO. By manually inspecting the ontology we observed that it includes the negative clause $\bot \leftarrow \mathsf{region}(x) \wedge \mathsf{junction}(x)$ and $Q(x) \leftarrow \mathsf{region}(x)$ is not UCQ-rewritable due to the following clauses in $\mathcal{T}$ (hence Lemma 9 fails):

$$\begin{aligned} \mathsf{region}(x) &\leftarrow \mathsf{engineered\_region}(x), \\ \mathsf{engineered\_region}(x) &\leftarrow \mathsf{region}(x) \wedge \mathsf{has\_origin}(x, y) \\ &\quad \wedge \mathsf{engineered\_region}(y) \end{aligned}$$

Our results for the rest of the ontologies are depicted in Tables 2 and 3. The former regards the process of computing a negative closure, which is query independent, and the latter the construction of IAR-rewritings for our test queries. In these tables column $t_\mathcal{R}$ presents the time required by Rapid to compute a rewriting $\mathcal{R}$, t_G the time our system required to check whether it can compute the negative closure using Grind and Lemma 9, and comlumn $t_{\mathcal{T}_{cn}}$ the time required to construct $\mathcal{T}_{cn}$; all times are in milliseconds. Moreover, $|\mathcal{T}_n|$ presents the number of negative clauses in the input TBox, $|\mathcal{T}_{cn}|$ the number of clauses in the negative closure constructed by our system, $|\mathcal{R}^{\mathsf{ir}}|$ the number of clauses in $\mathcal{R}^{\mathsf{ir}}$, column $\#q^\neg$ presents the number of queries in $\mathcal{R}^{\mathsf{ir}}$ that contain negative conjuncts and, finally, columns max and avg the maximum and average number of negative conjuncts in any clause in $\mathcal{R}^{\mathsf{ir}}$ with a negative part.

$\mathcal{T}$	t_G	$t_{\mathcal{T}_{cn}}$	$\lvert\mathcal{T}_n\rvert$	$\lvert\mathcal{T}_{cn}\rvert$
envo	16 452	166	5	124
FBbi	9 305	63	4	57
mohse	55 814	21	3	3
NBO	27 674	40	5	20
Not-Galen	63 424	30	3	11
XP	8 626	28	2	8
BFO	21 903	2 166	44	622
caro	28 759	4 232	82	1 043
Dolce-Lite	16 039	8 202	18	1 952

Table 2: Results for computation of negative closure; computation time (in msec) and sizes.

As can be seen for all ontologies we were able to check IAR-rewritability and then compute $\mathcal{T}_{cn}$ in a matter of few seconds up to a little over than a minute. Since this process only depends on the TBox and not the query, it can be done only once in an off-line step. Consequently, we feel that these times are quite encouraging.

Regarding the size of $\mathcal{T}_{cn}$, it did not increase significantly for ontologies mohse, nbo, Not-Galen, and XP, however, it did for envo, FBbi, BFO, caro and Dolce-Lite. This is because, although there are few negative clauses in the input ontology these involve concepts that have many sub-concepts in the ontology and hence many new negative clauses are implied creating an increase in the size of $\mathcal{T}_{cn}$. For example, in envo, two classes involved in a negative clause have 6 and 5 subclasses and this generates 41 new negative clauses, hence the size of $|\mathcal{T}_{cn}|$ is 25 times bigger than $|\mathcal{T}_n|$. However, as we will see next, since the size of the IAR-rewriting mostly depends on the concepts that appear in the query, this blow-up may not affect the final output.

Regarding the size of the IAR-rewriting ($|\mathcal{R}^{\mathsf{ir}}|$), it coincided with the size of the rewriting $\mathcal{R}$ for all ontologies but Dolce-Lite. In that ontology the sizes of $\mathcal{R}$ before extending with negative atoms were 4, 93, 45, 9, and 15 clauses for queries 1 to 5, respectively. These differences were due to the procedure saturate which introduces new clauses by applying variable unifications over the clauses of $\mathcal{R}$. Clearly, this can cause a significant increase in the size of the IAR-rewriting. In order to avoid it we restricted its application to elements of $\mathcal{R}$ that contain roles that also

[1] http://www.obofoundry.org/ontology/caro

[2] http://www.ifomis.org/bfo/1.1

[3] http://www.loa.istc.cnr.it/old/DOLCE

	$t_{\mathcal{R}}$	$\lvert\mathcal{R}^{\mathsf{ir}}\rvert$	$q^{\neg}$	max	avg
envo	94	765	33	7	6.5
	195	758	32	15	6.8
	163	768	33	7	6.5
	116	771	32	7	6.5
	110	765	35	16	7.3
FBbi	8	7	7	13	4.1
	7	11	11	4	2.2
	52	306	22	14	3.6
	6	11	11	7	4.4
	76	5	5	10	2.8
mohse	1 016	3 571	17	2	1.1
	10	3	1	2	2.0
	1 020	3 520	15	2	1.1
	1 049	3 511	13	2	1.1
	1 063	3 519	14	2	1.1
NBO	9	3	3	9	6.0
	9	2	2	6	4.0
	10	4	4	2	2.0
	205	1 328	14	4	1.8
	245	1 350	11	4	1.9

	$t_{\mathcal{R}}$	$\lvert\mathcal{R}^{\mathsf{ir}}\rvert$	$q^{\neg}$	max	avg
Not-Galen	37 456	16 684	19	6	1.6
	42 303	16 634	19	6	1.6
	55	28	7	6	1.7
	28 491	16 578	20	6	1.8
	13	4	3	2	2.0
XP	250	2 035	10	4	2.8
	335	2 115	12	4	2.6
	249	2 035	9	4	2.6
	333	2 037	9	4	2.7
	281	2 035	10	6	3.0
BFO	25	39	39	35	32.7
	9	7	7	65	38.4
	11	8	8	60	36.1
	8	9	9	95	40.1
	7	5	5	35	33.6
caro	10	2	2	91	68.5
	16	3	3	90	60.0
	154	40	40	53	43.9
	9	6	5	46	45.2
	7	8	7	46	45.0
Dolce-Lite	6	7	7	101	87.9
	21	198	198	258	94.0
	15	100	95	101	78.0
	11	19	19	121	101.7
	10	23	23	128	85.5

Table 3: Results for computation of IAR-rewritings.

appear in some negative clause. For example, it is not applied on $\mathcal{Q} = Q(x) \leftarrow A(x) \wedge R(x, y)$ if R does not occur in any negative clause of $\mathcal{T}_{cn}$. This is a quite effective optimisation since in practice ontologies rarely contain negative clauses that involve (either directly or via entailments) concepts with roles. That was indeed the case in all except just the Dolce-Lite ontology. Finally, the number of negative conjuncts added to clauses was in more than half of the cases quite small (less than 7 on average). In contrast, in three ontologies the algorithm had to add from 30 up to 100 negative conjuncts which could be a large number although the evaluation in (Tsalapati et al. 2016) showed that database and triple-store systems can cope with a fairly large number of negative atoms (even more than one hundred); further work in that respect is required to design optimisations that would reduce these numbers.

Summarising, our evaluation verifies the following non-trivial arguments:

- The condition we have described is usually satisfied in practice for a given TBox even if this is expressed in a DL for which the problem is intractable. Consequently, a negative closure for these TBoxes exists and an IAR-rewriting can be constructed using Algorithm 1.
- If a negative closure exists then computing it can be done relatively efficiently especially taking into account that the process is query independent and can be conducted only once at a pre-processing step.
- The number of clauses in the IAR-rewriting was in the vast majority of cases the same as that of the normal rewriting. This was due to the restrictions in the application of saturate which turned out to be very effective for real-world ontologies.
- The number of negative conjuncts added to the clauses was in most cases quite small, however, in some cases quite a few were added.

Conclusions

In this work we have studied the problem of query answering over DL-ontologies under the inconsistency-tolerant IAR semantics. First, we designed a general algorithm that can be applied on arbitary inputs but it may not terminate. We then defined a condition that ensures its termination and showed that this condition is satisfied by the relatively expressive DL DL-Lite$_{\mathsf{bool}}$ obtaining the first ever tractability result for IAR-answering over a non-Horn-DL, as well as, for semi-acyclic-$\mathcal{EL}$-TBoxes. Finally, we have provided a prototype implementation and preliminary evaluation obtaining encouraging results. More precisely, for almost all test ontologies and queries we were able to compute an IAR-rewriting within a reasonable time. This constitutes the first attempt towards IAR-aswering in the case of DL-ontologies that do not fall into the DL-Lite fragment.

Acknowledgements

Research supported by EU's Horizon 2020 research and innovation programme, grant agreement No. 720270 (HBP SGA1) and by the Research Centre of the Athens University of Economics and Business (Action 2, 2016-2017). Large part of this work was conducted when Giorgos Stoilos was working at AUEB.

References

Arenas, M.; Bertossi, L.; and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the Eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 68–79. ACM.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The DL-Lite Family and Relations. *Journal of Artificial Intelligence Research* 36:1–69.

Baader, F.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F. 2002. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.

Bertossi, L. E. 2006. Consistent query answering in databases. *SIGMOD Record* 35(2):68–76.

Bienvenu, M., and Rosati, R. 2013. New Inconsistency-Tolerant Semantics for Robust Ontology-Based Data Access. In *Proceedings of the Twenty-Sixth International Workshop on Description Logics*.

Bienvenu, M.; ten Cate, B.; Lutz, C.; and Wolter, F. 2014. Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. *ACM Transactions on Database Systems* 39(4):33:1–33:44.

Bienvenu, M.; Lutz, C.; and Wolter, F. 2012. Deciding FO-Rewritability in $\mathcal{EL}$. In *Proceedings of the Twenty-Fifth International Workshop on Description Logics*.

Bienvenu, M.; Lutz, C.; and Wolter, F. 2013. First-Order Rewritability of Atomic Queries in Horn Description Logics. In *Proceeding of the Twenty-Third International Joint Conference on Artificial Intelligence*.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *Journal of Automated Reasoning* 39(3):385–429.

Cuenca Grau, B.; Motik, B.; Stoilos, G.; and Horrocks, I. 2013. Computing Datalog Rewritings beyond Horn Ontologies. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*.

Eiter, T.; Ortiz, M.; Simkus, M.; Tran, T.-K.; and Xiao, G. 2012. Query Rewriting for Horn-$\mathcal{SHIQ}$ plus Rules. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Hansen, P.; Lutz, C.; Seylan, I.; and Wolter, F. 2015. Efficient Query Rewriting in the Description Logic $\mathcal{EL}$ and Beyond. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 3034–3040.

Hustadt, U.; Motik, B.; and Sattler, U. 2007. Reasoning in Description Logics by a Reduction to Disjunctive Datalog. *Journal of Automated Reasoning* 39(3):351–384.

Kaminski, M., and Grau, B. C. 2013. Sufficient Conditions for First-Order and Datalog Rewritability in $\mathcal{ELU}$. In *Proceedings of the Twenty-Sixth International Workshop on Description Logics*.

Kikot, S.; Kontchakov, R.; and Zakharyaschev, M. 2012. Conjunctive Query Answering with OWL 2 QL. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2010. Inconsistency-tolerant Semantics for Description Logics. In *Proceedings of Fourth International Conference on Web Reasoning and Rule Systems*, 103–117. Springer.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2011. Query rewriting for inconsistent DL-Lite ontologies. In *Proceedings of the Fifth International Conference on Web Reasoning and Rule Systems*, 155–169. Springer.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2015. Inconsistency-tolerant Query Answering in Ontology-Based Data Access. *Journal of Web Semantics* 33:3–29.

Pérez-Urbina, H.; Motik, B.; and Horrocks, I. 2010. Tractable Query Answering and Rewriting under Description Logic Constraints. *Journal of Applied Logic* 8(2):186–209.

Rosati, R. 2011. On the Complexity of Dealing with Inconsistency in Description Logic Ontologies. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 1057–1062.

Trivela, D.; Stoilos, G.; Chortaras, A.; and Stamou, G. 2015. Optimising Resolution-Based Rewriting Algorithms for OWL Ontologies. *Journal of Web Semantics* 33:30–49.

Tsalapati, E.; Stoilos, G.; Stamou, G. B.; and Koletsos, G. 2016. Efficient Query Answering over Expressive Inconsistent Description Logics. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 1279–1285.