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 EL_{\bot}, fragment of EL. 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_{\bot}, 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; P´erez-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 EL to 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 EL and even Horn-SHIQ. To provide a positive result in terms of tractable data complexity Rosati defined 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-EL (Bienvenu, Lutz, and Wolter 2012) as well as in DL-Lite_{\bot} (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 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
the same arity. A tuple of constants $\bar{a}$ is a certain answer of $Q$ over a KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ if the arity of $\bar{a}$ agrees with the arity of $Q$ and $\mathcal{T} \cup \mathcal{A} \models Q(\bar{a})$, where $Q(\bar{a})$ denotes the boolean query obtained by replacing all answer variables in $Q$ with $\bar{a}$. We use $\text{cert}(Q, \mathcal{T} \cup \mathcal{A})$ to denote all certain answers of $Q$ w.r.t. $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$.

**Definition 1.** Let $\mathcal{T}$ be an $\mathcal{L}$-TBox, $\mathcal{A}$ an ABox consistent w.r.t. $\mathcal{T}$ and $Q$ a CQ. A disjunctive datalog-rewriting (or simply rewriting) of $Q$ w.r.t. $\mathcal{T}$ is a disjunctive datalog program $\mathcal{R}$ such that $\mathcal{T} \cup \mathcal{A} \models Q(\bar{a})$ iff $\mathcal{R} \cup \mathcal{A} \models Q(\bar{a})$, or in case $\mathcal{Q}$ is boolean $\mathcal{T} \cup \mathcal{A} \models Q$ iff $\mathcal{R} \cup \mathcal{A} \models Q$. We say that a query $Q$ is (disjunctive) datalog-rewritable w.r.t. $\mathcal{T}$ if there exists a (disjunctive) datalog-rewriting $\mathcal{R}$ of $Q$ w.r.t. $\mathcal{T}$; if $\mathcal{R}$ is a UCQ, then $Q$ 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 $\mathcal{E}_{\perp}$ and Horn-SHIQ a datalog-rewriting (Eiter et al. 2012). Disjunctive programs are required in the case of highly expressive DLs like 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 $\mathcal{A}_r$ 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 $\bar{a}$ is called an IAR-answer of $\mathcal{Q}$ over $\mathcal{K}$ if $\bar{a} \in \text{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}_r)$. We use $\text{cert}_r(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$ to denote the set of all IAR-answers of $\mathcal{Q}$ over $\mathcal{K}$ and we also write $\mathcal{T} \cup \mathcal{A} \models_{\text{ir}} \mathcal{Q}(\bar{a})$.

**A Framework for IAR-answering**

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

<table>
<thead>
<tr>
<th>DL Axiom</th>
<th>Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{E}_{\perp}$ and DL-Lite</td>
<td></td>
</tr>
<tr>
<td>$B_1 \sqcap B_2 \subseteq A$</td>
<td>$A(x) \leftarrow B_1(x) \land B_2(x)$</td>
</tr>
<tr>
<td>$A \subseteq \exists R.B$</td>
<td>$R(x, f(x)) \leftarrow A(x), B(f(x)) \leftarrow A(x)$</td>
</tr>
<tr>
<td>$\exists R \subseteq A$</td>
<td>$A(x) \leftarrow R(x, y)$</td>
</tr>
<tr>
<td>$A \sqcap B \subseteq \bot$</td>
<td>$\bot \leftarrow A(x) \land B(x)$</td>
</tr>
<tr>
<td>$\mathcal{E}_{\perp}$</td>
<td></td>
</tr>
<tr>
<td>$\exists R.B \subseteq A$</td>
<td>$A(x) \leftarrow R(x, y) \land B(x)$</td>
</tr>
<tr>
<td>$P \subseteq R$</td>
<td>$R(x, y) \leftarrow P(x, y)$</td>
</tr>
<tr>
<td>$P \subseteq \neg R$</td>
<td>$\bot \leftarrow R(x, y) \land P(x, y)$</td>
</tr>
<tr>
<td>DL-Lite$_{\text{bool}}$</td>
<td></td>
</tr>
<tr>
<td>$A \subseteq B_1 \sqcup B_2$</td>
<td>$B_1(x) \lor B_2(x) \leftarrow A(x)$</td>
</tr>
<tr>
<td>$\exists R. \text{self} \subseteq A$</td>
<td>$A(x) \leftarrow R(x, y)$ (dually with $A \subseteq \exists R. \text{self}$)</td>
</tr>
</tbody>
</table>

Table 1: Translation of DL axioms into FOL.
Example 1. Let the TBox $T = \{\bot \leftarrow A(x) \wedge C(x), A(x) \leftarrow P(x, y)\}$, the ABox $A = \{C(a), P(a, c), A(b)\}$ and the CQ $Q = Q(x) \leftarrow A(x)$. Clearly, $A$ is inconsistent w.r.t. $T$ and the repairs are $A_r_1 = \{C(a), A(b)\}$ and $A_r_2 = \{P(a, c), A(b)\}$. Hence, we have $A_r = \{A(b)\}$ and $\text{cert}(Q, T \cup A_r) = \{\} = \text{cert}_{A}(Q, T \cup A)$. Instead, consider the rewriting $R = \{Q(x) \leftarrow A(x), Q(x) \leftarrow P(x, y)\}$ of $Q$ w.r.t. $T$. We can note that $\text{cert}(R, A) = \{a, b\}$ 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 $R$ with negative atoms which will prevent the clauses of $R$ to bind to certain patterns of $A$. For example, due to $T \models \bot \leftarrow A(x) \wedge C(x)$ and $T \models \bot \leftarrow P(x, y) \wedge C(x)$, $R$ should be extended to $R^\sim = \{Q(x) \leftarrow A(x) \wedge \neg C(x), Q(x) \leftarrow P(x, y) \wedge \neg C(x)\}$. Then, $\text{cert}(R^\sim, A) = \{\}$. 

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

Definition 3. Given an $\mathcal{L}$-TBox and a CQ $Q$, an IAR-rewriting $R^\prime$ of $Q$ w.r.t. $T$ is a disjunctive datalog program, possibly extended with negation symbols, such that for every ABox $A$ we have $T \cup A \models_{\mathcal{A}} \text{Q}(\bar{x})$ iff $R^\prime \cup A \models_{\mathcal{A}} \text{Q}(\bar{x})$.

Example 1 suggests that to compute an IAR-rewriting we should at least compute all possible negative clauses $C$ such that $T \models C$ 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 $T$.

Definition 4. A negative closure of a $\mathcal{L}$-TBox $T$, denoted by $T_{cn}$, is a finite set of negative clauses such that $T \models \bot \leftarrow \bigwedge \beta_i$ iff some $\bot \leftarrow \bigwedge \alpha_i$ in $T_{cn}$ 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 $C$ should be considered, only if no other clause is a syntactical subset of $C$. 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 $Q = Q(x) \leftarrow A(x)$, the TBox $T_1 = \{C\}$, where $C \models \bot \leftarrow A(x) \wedge B(x) \wedge C(x)$ and the $T_1$-inconsistent ABox $A = \{A(a), B(a), C(a)\}$. There are three repairs, $\{A(a), B(a)\}, \{A(a), C(a)\}$, and $\{B(a), C(a)\}$, hence $A_r = \emptyset$ and $\text{cert}_{A}(Q, T_1 \cup A) = \emptyset$.

Following the technique in (Lembo et al. 2011) consider the rewriting $R = \{Q(x) \leftarrow A(x)\}$ of $Q$ w.r.t. $T_1$. Due to $C \in T_1$, $R$ should be extended to $R^\sim = \{Q(x) \leftarrow A(x) \wedge \neg(B(x) \wedge C(x))\}$ for which we have $\text{cert}(R^\sim, A) = \emptyset$ as required.

### Algorithm 1 IAR-Rewriting

**Input:** a CQ $Q$ and an $\mathcal{L}$-TBox $T$

1. Compute a negative closure $T_{cn}$ of $T$
2. $T_{cn} := \text{minimise}(\text{saturation}(T_{cn}))$
3. Compute a rewriting $R$ of $Q$ w.r.t. $T$
4. $R^\prime := \text{saturation}(R)$
5. $R^\prime := \emptyset$
6. for $Q \in R^\prime$ do
7. $R^\prime := \emptyset$
8. for $\alpha \in Q$ where $\alpha$ is not an inequality atom do
9. for $\bot \leftarrow \beta_1 \land \ldots \land \beta_n$ in $T_{cn}$ do
10. if $\alpha = \beta_k$ then
11. $\text{Add } (\neg(\beta_1 \land \ldots \land \beta_m) \mu) \text{ to } Q^\prime$
12. end if
13. end for
14. end for
15. $R^\prime := R^\prime \cup \{Q^\prime\}$
16. end for
17. return $R^\prime$

Now consider the TBox $T_2 = T_1 \cup \{A(x) \leftarrow B(x)\}$. Interestingly, $\{B(a), C(a)\}$ is no longer a repair of $A$ and hence $A_r = \{A(a)\}$. Therefore, $\text{cert}_{A}(Q, T_2 \cup A) = \{a\}$ and to construct a correct IAR-rewriting of $Q$ w.r.t. $T_2$ one should not add $(\neg(B(x) \wedge C(x)))$ to $Q(x) \leftarrow A(x) \in R$. Notice that in the new TBox we have $T_2 \models C$, where $C = \bot \leftarrow B(x) \wedge C(x)$ and, moreover, $C \models C$ is a syntactical subset of $C$.

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

$$
\begin{align*}
Q_1 &= Q(x) \leftarrow R(x, y) \land x \neq y \\
Q_2 &= Q(x) \leftarrow R(x, y) \land \neg S(x, x)
\end{align*}
$$

Consider the ABoxes $A_1 = \{R(a, b), S(a, b)\}$, $A_2 = \{R(a, a), S(a, a)\}$. As expected, $\text{cert}(R^\sim, A_1) = \{a\}$ and $\text{cert}(R^\sim, A_2) = \emptyset$. Note that we cannot drop the conjunct $x \neq y$ from $Q_1$ as then we would incorrectly have $\text{cert}(R^\sim, A_2) = \{a\}$.

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 notion of distinct variables illustrated in Example 3. Roughly speaking given a clause $C$, this procedure replaces it with a list of clauses $C_1, \ldots, C_n$ such that $C$ is equivalent to $C_1 \lor \ldots \lor C_n$ and for every $C_i$ if $x, y$ are pairs of distinct variables then $C_i$ contains the conjunct $x \neq y$. This list of clauses is generated by using unification on the variables of $C$, hence $n$ can be exponential in the number of variables in $C$. Finally, minimise is defined as follows:
**Definition 5.** Let $T$ be a set of negative clauses. Procedure minimise$(T)$ returns a new set of clauses $T'$ such that $T' \models T$ and for every $C \in T'$ no $C'$ in $T'$ different to $C$ is (up to variable renaming) a syntactical subset of $C$.

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 $EL$-TBox $T$  

\[
\begin{align*}
d(x) \land A(x) \\
A(x) &\leftarrow R(x, y) \land B(y)
\end{align*}
\]

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

\[
R = \{Q(x) \leftarrow A(x), A(x) \leftarrow R(x, y) \land B(y)\}
\]

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

\[
R^{ir} = \{Q(x) \leftarrow A(x) \land \neg (A(x) \land d(x)), A(x) \leftarrow R(x, y) \land x \neq y \land B(y) \land \neg (R(x, y) \land x \neq y \land B(y) \land d(x)), A(x) \leftarrow R(x, x) \land B(x) \land \neg (R(x, x) \land B(x) \land d(x))\}
\]

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

**Proof.** (sketch) If there exists a rewriting $R$ of $Q$ w.r.t. $T$, and a negative closure $T_{cn}$, then Algorithm 1 terminates. The output $R^{ir}$ is a disjunctive datalog program possibly extended with negative atoms according to lines 5–16. To prove correctness of Algorithm 1 we show that $R \cup A \models R^{ir}$, $Q(\overline{a})$ iff $R^{ir} \cup A \models Q(\overline{a})$; this is done using induction on the evaluation of $R^{ir}$ ($R$) over $A$. 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 $Q = Q(x) \leftarrow B(x)$ and let also the following $EL_\bot$-TBox $T$:

\[
\begin{align*}
\bot &\leftarrow A(x) \land B(x) \\
A(x) &\leftarrow R(x, y) \land A(y)
\end{align*}
\]

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

Intuitively, the main reason for non-existence of $T_{cn}$ 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 $EL$-TBox $T$. We say that $ABox$-inconsistency is UCQ-rewritable relative to $T$ if a union of boolean queries $R$ with head atom $Q$ exists s.t. for every ABox $A$,$A$ is inconsistent w.r.t. $T$ iff $A \cup R \models Q$. $R$ is an UCQ-rewriting of $ABox$-inconsistency relative to $T$.

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

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

Let $T_{cn}$ be constructed from $R$ as described above and let $T \models C$ for some clause $C = \bot \leftarrow \bigwedge_{i=1}^{n} \beta_{i}$. For $\sigma$ an injective instantiation of the variables of $C$ we have that $T \cup C \models \bot$ or $T \cup \bigwedge_{i=1}^{n} \beta_{i} \models \bot$, i.e., $\\{ \beta_{1}, \ldots, \beta_{n} \}$ is inconsistent. Then, some CQ $Q = Q \leftarrow \bigwedge_{i=1}^{m} \alpha_{i}$ must exist such that $\{ \beta_{1}, \ldots, \beta_{n} \} \cup \{ Q \} \models Q$. Since $Q$ does not appear anywhere in $T$ this implies that some mapping $\mu$ from the variables of $Q$ to individuals in $\{ \beta_{1}, \ldots, \beta_{n} \}$ exists such that we have $\{ \alpha_{1}, \ldots, \alpha_{m} \} \subseteq \{ \beta_{1}, \ldots, \beta_{n} \}$. Since $\sigma$ is injective we can compute its inverse $\sigma^{-1}$; then we have $\{ \alpha_{1}\sigma^{-1}, \ldots, \alpha_{m}\sigma^{-1} \} \subseteq \{ \beta_{1}\sigma^{-1}, \ldots, \beta_{n}\sigma^{-1} \}$ or $\{ \alpha_{1}\sigma^{-1}, \ldots, \alpha_{m}\sigma^{-1} \} \subseteq \{ \beta_{1}, \ldots, \beta_{n} \}$. Consequently, some $\lambda = \mu\sigma^{-1}$ exists such that $\{ \alpha_{1}, \ldots, \alpha_{m} \} \subseteq \{ \beta_{1}, \ldots, \beta_{n} \}$. By construction, $T_{cn}$ contains a clause of the form $\bot \leftarrow \bigwedge_{i=1}^{m} \alpha_{i}$ which by the above we have shown that it subsumes $C$. Moreover, $T_{cn}$ is finite since $R$ is finite.

For the opposite direction, from a negative closure $T_{cn}$ we can construct a UCQ-rewriting for $ABox$-inconsistency relative to $T$ by replacing the head atoms of clauses in $T_{cn}$ 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 $T$ be an $EL$-TBox where $L$ is a Horn-DL. Let the set of $L$-concepts $S = \{ A_i(x) \mid \bot \leftarrow A_i(x) \land \ldots \land A_i(x) \}$.  

$1976$
Algorithm 1 constructs: can use Lemma 9 to show the following.

- UCQ-rewritable. Moreover, queries over so called semi-acyclic-


\[ T \] is UCQ-rewritable w.r.t. \( S \) if \( T \) is UCQ-rewritable relative to \( S \).

Example 6. Let the DL-Lite\textsubscript{bool}-TBox \( T = \{ \bot \leftarrow R(x, y) \land A(x), A(x) \lor D(x) \leftarrow C(x) \} \) and the instance query \( Q(x) \leftarrow D(x) \) (by assumption this rewriting exists). Finally add to \( T \) the clause \( \bot \leftarrow R_1(x) \land \ldots \land R_m(x) \).

Interestingly, in the case of the non-Horn DL-Lite\textsubscript{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\textsubscript{bool}-TBox. Therefore, these two results together with Lemma 9 can be used to show the first ever positive result on IAQ-rewritability for a non-Horn DL.

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

**Proof.** (sketch) Let \( T \) be an arbitrary DL-Lite\textsubscript{bool}-TBox. By applying the procedure of Cuenca Grau et al. (2013) \( T \) 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\textsubscript{bool} clauses we consider here). Moreover, by the results in (Artale et al. 2009) every instance query formed using symbols of \( T \) is UCQ-rewritable hence Lemma 9 can be applied.

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

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

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

Restricting \( E\mathcal{L}_{\bot} \) to obtain a fragment for which IAR-answering is tractable (w.r.t. data complexity) was also studied by Rosati (2011) who defined \( E\mathcal{L}_{\bot,\bot} \). 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 \( E\mathcal{L}_{\bot,\bot} \) using our framework. For a better comparison with the original definition and conciseness in the following we use DL notation.

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

Intuitively, if such \( \sigma \) 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 \( C' \sqsubseteq C \).

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

For arbitrary Horn-DLs that are not always UCQ-rewritable (like, e.g., \( E\mathcal{L} \)) 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 \( E\mathcal{L} \), \( E\mathcal{L}_{\bot} \) and Horn-SHIT\( \mathcal{F} \) 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-ELIU (Kaminski and Grau 2013) a fragment of 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 ELIU 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-ELIU is IAR-rewritable.

### Evaluation

Based on Algorithm 1 we created a prototype system. It is using Rapid (Trivelà et al. 2015) to compute a rewriting \( R \) for \( Q \) and \( T \) (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 \( T_{cn} \). If the negative closure can be computed, then our system proceeds in extending \( R \) 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 ELIU 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 ELIU fragments of the ontologies CARO, BFO and Dolce-Lite. 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 \( B \subseteq \neg C \) we have constructed queries \( Q(x) \leftarrow A(x) \) and \( Q(x) \leftarrow D(x) \) such that \( T \models A \subseteq B \) and \( T \models B \subseteq D \). 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 \( \bot \leftarrow \text{region}(x) \land \text{junction}(x) \) and \( Q(x) \leftarrow \text{region}(x) \) is not UCQ-rewritable due to the following clauses in \( T \) (hence Lemma 9 fails):

\[
\text{region}(x) \leftarrow \text{engineered-region}(x),
\]
\[
\text{engineered-region}(x) \leftarrow \text{region}(x) \land \text{has-origin}(x, y) \\
\land \text{engineered-region}(y)
\]

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 \( t_R \) presents the time required by Rapid to compute a rewriting \( R \), \( t_G \) the time our system required to check whether it can compute the negative closure using Grind and Lemma 9, and column \( t_{cn} \), the time required to construct \( T_{cn} \); all times are in milliseconds. Moreover, \(|T_{cn}|\) presents the number of negative clauses in the input TBox, \(|T_{cn}|\) the number of clauses in the negative closure constructed by our system, \(|\mathcal{R}^n|\) the number of clauses in \( \mathcal{R}^n \), column \#q− presents the number of queries in \( \mathcal{R}^n \) that contain negative conjuncts and, finally, columns max and avg the maximum and average number of negative conjuncts in any clause in \( \mathcal{R}^n \) with a negative part.

As can be seen for all ontologies we were able to check IAR-rewritability and then compute \( T_{cn} \) 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 \( T_{cn} \), 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 \( T_{cn} \). 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 \(|T_{cn}|\) is 25 times bigger than \(|T_n|\). 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 \(|\mathcal{R}^n|\), 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

| \( \mathcal{T} \) | \( t_R \) | \( t_G \) | \(|T_n|\) | \(|T_{cn}|\) |
| --- | --- | --- | --- | --- |
| 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.
appear in some negative clause. For example, it is not applied on $Q = Q(x) \leftarrow A(x) \land R(x,y)$ if $R$ does not occur in any negative clause of $T_{cn}$. 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 (Tsapalati 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 arbitrary 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$_b$ obtaining the first ever tractability result for IAR-answering over a non-Horn-DL, as well as, for semi-acyclic-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-answering 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.

### Table 3: Results for computation of IAR-rewritings.

| | $t_{cn}$ | $|\mathcal{R}|$ | $q$ | 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 | 2.2 |
| 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 |
| 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 | 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 |
| BFO | 10 | 2 | 2 | 91 | 68.5 |
| | 16 | 3 | 3 | 90 | 60.0 |
| | 154 | 40 | 40 | 53 | 43.9 |
| | 9 | 6 | 6 | 46 | 45.2 |
| | 7 | 8 | 7 | 46 | 45.0 |
| caro & 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 |
References


