Querying expressive DL Ontologies under the ICAR semantics

Despoina Trivela[1], Giorgos Stoilos[2], and Vasilis Vassalos[1].

1 : Athens University of Economics and Business, Athens, Greece.
2 : Babylon Health London, SW3 3DD, UK.

Abstract. Inconsistency-tolerant semantics, like the ICAR semantics, have been proposed to perform query answering over inconsistent DL knowledge bases. In the current paper we propose a general framework for ICAR-answering over arbitary Horn-DLs that is based on rewriting. We describe conditions for termination of our rewriting algorithm and show that existing techniques and results on UCQ-rewritability can be used to check if they are satisfied for a given input ontology and query.

1 Introduction

Query answering over ontologies expressed in Description Logics (DL) has recieved significant attention from both a practical and a theoretical perspective. In the vast majority of works, the problem has been studied over datasets that are consistent w.r.t. the ontology [9, 17, 12, 20]. However, in real-world applications this is not always the case, e.g. in data integration applications the data deriving from disparse sources may contradict the ontological axioms. A straightforward approach to perform consistent query answering would be to first remove the conflicting elements from the datasets. However, this is not always possible as the data may be reside in distributed or access restricted data sources, or be subject to frequent and diverse modifications.

To address this issue, inconsistency-tolerant semantics have been proposed that describe which answers are meaningful to be returned in the presence of inconsistent data. Examples of such semantics are the IAR, ICAR, AR semantics [15, 14] that are based on the notion of the *repair*, that is a maximal (w.r.t. inclusion) consistent subset of the original dataset. The IAR and ICAR semantics have shown to have better computational properties since the query evaluation problem over DL-Lite ontologies is in AC^0 w.r.t. data complexity [16], while it is in coNP for the AR semantics. However, computing answers using inconsistency-tolerant semantics has been proved quite difficult for DLs more expressive than DL-Lite. More precisely, Rosati [18] showed that the problem of IAR and ICAR-answering is at least coNP-hard w.r.t. data complexity for almost all well-known DLs from $\mathcal{EL}_\bot$ to $\mathcal{SHIQ}$. Moreover, in the $\mathcal{EL}_{\bot nr}$ fragment of $\mathcal{EL}_\bot$ where query answering is tractable for the IAR semantics, the problem remains in coNP for the ICAR semantics. Consequently, important research results on consistent query answering [18, 7, 16, 3, 22] focus on fragments of DL-Lite. However, in [22]

a practical system was proposed for the ICAR and IAR semantics that computes upper approximations for DLs more expressive than DL-Lite. Moreover, an algorithm for IAR-answering was proposed in [21] that can handle arbitary DLs but need not terminate. Despite the work presented in [22] the problem of designing practical ICAR-answering algorithms for expressive DLs is open.

In the current paper we extend our previous work of [21] and study ICAR-answering over expressive DL ontologies. More precisely, we present a general framework for ICAR-answering that is based on query rewriting. Our rewriting algorithm has as a starting point the one presented in [15]. Given an input query and ontology expressed in a Horn-DL if our algorithm terminates, it computes a datalog program, extended with negation, that can be evaluated over the initial dataset to compute the ICAR-answers. Based on our analysis and previous results [21] we describe the conditions that ensure termination of our algorithm. The termination conditions are related to the notion of UCQ-rewritability that has been studied quite extensively in DLs [1, 4, 11]. Consequently, we are able to provide positive results for instance queries and ontologies expressed in semi-acyclic-$\mathcal{EL}_\bot$ [5], as well as UCQ-rewritable queries over ontologies expressed in $\mathcal{EL}_{\bot nr}$. For DLs and queries for which the termination conditions are not generally satisfied, we exploit previous works [6, 8, 11] and provide an approach to check termination over a fixed input ontology and query. This allows us to design a framework for ICAR-answering over expressive Horn-DLs. Finally, we have conducted a preliminary evaluation of our approach. We obtained positive results showing that it is possible to perform ICAR-answering even in the case of DLs for which the problem is in general intractable.

2 Preliminaries

Description Logics A DL *knowledge base* (KB) $\mathcal{K}$ consists of a TBox $\mathcal{T}$, and an ABox $\mathcal{A}$, $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$. $\mathcal{T}$ and $\mathcal{A}$ are constructed from the countable and pairwise disjoint sets $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{I}$ of atomic concepts (unary predicates), atomic roles (binary predicates), and individuals (constants). 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 TBox $\mathcal{T}$ is a set of DL axioms. An $\mathcal{EL}_\bot$ concept is inductively defined by the syntax: $C := \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid \exists R.C$, where $A \in \mathbf{C}$ and $R \in \mathbf{R}$ 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$_R$ (or simply DL-Lite) restricts $\mathcal{EL}_\bot$ by allowing concepts of the form A, and $\exists R.\top$; R in DL-Lite can also be the *inverse* of a role of the form S^- and we can also have role inclusions of the form $S \sqsubseteq R$ or $S \sqsubseteq \neg R$ for S, R roles. An ABox $\mathcal{A}$ is *consistent* w.r.t. some TBox $\mathcal{T}$ if there exists a model for the KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$; otherwise it is *inconsistent*. The semantics of DLs can be given by a well-known translation to First-Order Logic (FOL) [2]. Table 1 presents the translation of $\mathcal{EL}_\bot$ and DL-Lite axioms to first order clauses (inverse roles have been omitted). In the following we assume that the TBox axioms are translated into FOL.

Table 1: Translation of DL axioms into FOL

DL Axiom	**Clause**
$B \sqsubseteq A$	$A(x) \leftarrow B(x)$
$A \sqcap B \sqsubseteq \bot$	$\bot \leftarrow A(x) \wedge B(x)$
$B_1 \sqcap B_2 \sqsubseteq A$	$A(x) \leftarrow B_1(x) \wedge B_2(x)$
$A \sqsubseteq \exists R.B$	$R(x, f(x)) \leftarrow A(x), B(f(x)) \leftarrow A(x)$
$\exists R \sqsubseteq A$	$A(x) \leftarrow R(x, y)$
$\exists R.B \sqsubseteq A$	$A(x) \leftarrow R(x, y) \wedge B(x)$
$P \sqsubseteq R$	$R(x, y) \leftarrow P(x, y)$
$P \sqsubseteq \neg R$	$\bot \leftarrow R(x, y) \wedge P(x, y)$

Datalog and Conjunctive Queries A *disjunctive datalog clause* r (also called *rule*) 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 positive or negative atoms called the *body* of the clause and $\psi(\vec{x})$ is a disjunction of positive atoms called its *head*. For simplicity we will omit variable quantifiers and write $\psi(\vec{x}) \leftarrow \phi(\vec{x}, \vec{y})$. A *datalog clause* is a disjunctive datalog clause where the head contains a single atom. A *Horn-clause* is a datalog clause where the body contains only positive atoms. A *(disjunctive) datalog program* $\mathcal{P}$ is a finite set of (disjunctive) datalog clauses. We consider Herbrand models over all constants from $\mathcal{P}$. We say that a model M of $\mathcal{P}$ is *minimal* if there is no model M' of $\mathcal{P}$ such that M' is a subset of M. A positive ground atom $D(\vec{a})$ is entailed by $\mathcal{P}$ iff all minimal models of $\mathcal{P}$ contain $D(\vec{a})$; a negative ground atom $\neg D(\vec{a})$ is entailed by $\mathcal{P}$ iff $D(\vec{a})$ is not icluded in the minimal models of $\mathcal{P}$. The evaluation of $\mathcal{P}$ over an ABox $\mathcal{A}$ is the set of ground atoms entailed by the program $\mathcal{P} \cup \mathcal{A}$.

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 $Q(x) \leftarrow A(x)$ (we often simply write $A(x)$). A UCQ is a finite set of CQs. 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 the answer variables 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. $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$.

Definition 1. *Let $\mathcal{T}$ be a TBox and $\mathcal{Q}$ a CQ. A* datalog-rewriting *(or simply* rewriting*) of $\mathcal{Q}$ w.r.t. $\mathcal{T}$ is a datalog program $\mathcal{R}$ such that for any ABox $\mathcal{A}$ consistent w.r.t. $\mathcal{T}$ we have $\mathcal{T} \cup \mathcal{A} \models Q(\vec{a})$ iff $\mathcal{R} \cup \mathcal{A} \models Q(\vec{a})$, 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 $\mathcal{Q}$ is* datalog-rewritable *w.r.t. $\mathcal{T}$ if there exists a 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. $\mathcal{T}$.*

Note that we will refer to a clause of the form $H(\vec{s}) \leftarrow \bigwedge_i \alpha_i \wedge \bigwedge_j \neg \mathcal{B}_j$, where α_i are positive atoms and $\mathcal{B}_j$ are conjunctions of positive atoms, as a datalog clause. Indeed such a clause is equisatisfiable to a datalog program that inlcudes $H(\vec{s}) \leftarrow \bigwedge_i \alpha_i \wedge \bigwedge_j \neg \beta_j$ and $\beta_j \leftarrow \mathcal{B}_j$, for all j, where β_j are positive atoms.

Inconsistency-tolerant Semantics Definitions 2 and 3 recapitulate some of the notions used in the IAR and ICAR semantics [15]. Defintion 4 formalises the notion of the rewriting under the IAR and ICAR semantics.

Definition 2. *Consider a TBox $\mathcal{T}$ and ABox $\mathcal{A}$ we define the* consistent logical consequences of $\mathcal{T}, A$ *as the set $clc(\mathcal{T}, \mathcal{A}) = \{a \mid$ some $S \subseteq \mathcal{A}$ exists s.t. $\mathcal{T} \cup S \models a$ and S is consistent w.r.t. $\mathcal{T}\}$, where we use a to denote an assertion.*

Definition 3. *A* repair *of a set of assertions S w.r.t. a TBox $\mathcal{T}$ is any maximal (w.r.t. set inclusion) subset of S that is consistent w.r.t. $\mathcal{T}$.*

- *We use $\mathcal{A}_{\mathsf{ir}}$ to denote the* intersection *of all repairs of $\mathcal{A}$ w.r.t. $\mathcal{T}$. Let $\mathcal{Q}$ be a CQ and let $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ be a 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 $\mathcal{T} \cup \mathcal{A} \models_{\mathsf{ir}} \mathcal{Q}(\vec{a})$.*
- *We use $\mathcal{A}_{\mathsf{icar}}$ to denote the* intersection *of all repairs of $clc(\mathcal{T}, \mathcal{A})$. Let $\mathcal{Q}$ be a CQ and let $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ be a KB. A tuple of constants $\vec{a}$ is called a* ICAR-answer *of $\mathcal{Q}$ over $\mathcal{K}$ if $\vec{a} \in \mathsf{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}_{\mathsf{icar}})$. We use $\mathsf{cert}_{\mathsf{icar}}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$ to denote the set of all ICAR-answers of $\mathcal{Q}$ over $\mathcal{K}$ and we also write $\mathcal{T} \cup \mathcal{A} \models_{\mathsf{icar}} \mathcal{Q}(\vec{a})$.*

Definition 4. *Given a TBox and a CQ $\mathcal{Q}$, an* IAR-rewriting $\mathcal{R}^{\mathsf{ir}}$ *of $\mathcal{Q}$ w.r.t. $\mathcal{T}$ is a datalog program such that for every ABox $\mathcal{A}$ we have $\mathcal{T} \cup \mathcal{A} \models_{\mathsf{ir}} \mathcal{Q}(\vec{a})$ iff $\mathcal{R}^{\mathsf{ir}} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$. Similarly, for an* ICAR-rewriting $\mathcal{R}^{\mathsf{icr}}$ *we have $\mathcal{T} \cup \mathcal{A} \models_{\mathsf{icr}} \mathcal{Q}(\vec{a})$ iff $\mathcal{R}^{\mathsf{icr}} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$.*

3 Towards an ICAR-answering algorithm

The problem of answering queries under the ICAR semantics was first investigated in [15] where a rewriting approach was presented for DL-Lite. Given an input TBox and query, the proposed algorithm computes a rewriting of the query that can be evaluated over any ABox to obtain the ICAR-answers. Example 1 illustrates the approach of [15].

Example 1. Let $\mathcal{T}$ be the DL-Lite TBox $\mathcal{T} = \{C(x) \leftarrow A(x), \bot \leftarrow A(x) \wedge B(x)\}$, $\mathcal{Q}$ the query $\mathcal{Q} = Q(x) \leftarrow C(x)$ and $\mathcal{A}$ the ABox $\mathcal{A} = \{A(a), B(a)\}$. $\mathcal{A}$ has two repairs, that is $\{A(a)\}$ and $\{B(a)\}$, and hence $\mathcal{A}_{\mathsf{ir}} = \emptyset$. It is not hard to verify that $clc(\mathcal{T}, \mathcal{A}) = \{C(a), A(a), B(a)\}$. Moreover, $clc(\mathcal{T}, \mathcal{A})$ has two repairs, that is $\{A(a), C(a)\}$ and $\{B(a), C(a)\}$ and hence $\mathcal{A}_{\mathsf{icar}} = \{C(a)\}$. Therefore, $\mathsf{cert}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}_{\mathsf{icar}}) = \{a\}$ and by definition of the ICAR-answers it holds that $\mathsf{cert}_{\mathsf{icar}}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}) = \{a\}$.

In the first step, the algorithm in [15] computes the rewriting $\mathcal{R}$ of $\mathcal{Q}, \mathcal{T}$ under the standard semantics, $\mathcal{R} = \{Q(x) \leftarrow C(x), Q(x) \leftarrow A(x)\}$. Then, it extends the queries in $\mathcal{R}$ with the appropriate negative atoms, $\mathcal{R}' = \{Q(x) \leftarrow C(x), Q(x) \leftarrow A(x) \wedge \neg B(x)\}$. The negative atoms in $\mathcal{R}'$ guarantee that the evaluation of $\mathcal{R}'$ over $\mathcal{A}$ will only return answers from $\mathcal{A}_{\mathsf{ir}}$. Indeed, atom $\neg B(x)$ prevents $Q(x) \leftarrow A(x) \wedge \neg B(x)$ from binding with $A(a)$ which is not included

in $\mathcal{A}_{\mathsf{ir}}$. At next step, the algorithm applies the rewriting procedure (under the standard semantics) once more on the elements of $\mathcal{R}'$ (only on the positive atoms) to obtain $\mathcal{R}'' = \mathcal{R}' \cup \{Q(x) \leftarrow A(x)\}$ that captures the assertions in $clc(\mathcal{T}, \mathcal{A})$. When $Q(x) \leftarrow A(x)$ of $\mathcal{R}''$ is evaluated over $\mathcal{A}$ we obtain the ICAR-answer $\{a\}$, $\mathsf{cert}(\mathcal{R}'', \mathcal{A}) = \mathsf{cert}_{\mathsf{icar}}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A})$. ♢

A hybrid approach for ICAR-answering was presented in [22] that employs a rewriting, as well as an ABox saturation procedure. More precisely, given an input query $\mathcal{Q}$, a TBox $\mathcal{T}$, and an ABox $\mathcal{A}$, the algorithm in [22] exploits existing approaches [13, 19] to compute the saturated ABox, that is the set of the assertions entailed from $\mathcal{A}$ and the axioms of $\mathcal{T}$ that can be translated into datalog. Then, it evaluates the IAR-rewriting of $\mathcal{Q}, \mathcal{T}$ over the saturated $\mathcal{A}$. The algorithm supports DL-Lite and it can be used to compute upper approximations of the ICAR-answers for more expressive DLs.

Example 2. Consider the following $\mathcal{EL}_\bot$ TBox $\mathcal{T}$, the query $\mathcal{Q} = Q(x) \leftarrow C(x)$ and the ABox $\mathcal{A} = \{A(a), B(a)\}$.

$$\begin{aligned}\mathcal{T} = \{\ & C(x) \leftarrow A(x) \wedge K(x) \\ & K(x) \leftarrow B(x) \\ & \bot \leftarrow A(x) \wedge B(x)\}\end{aligned}$$

It is not hard to verify that $clc(\mathcal{T}, \mathcal{A}) = \{A(a), B(a), K(a)\}$ and that $\mathcal{A}_{\mathsf{icar}} = \{K(a)\}$. Hence, $\mathsf{cert}_{\mathsf{icar}}(\mathcal{Q}, \mathcal{T} \cup \mathcal{A}) = \emptyset$.

By applying the technique of [22] we first compute the saturation of $\mathcal{A}$, $\mathcal{A}_s = \{C(a), A(a),\ B(a), K(a)\}$ and the IAR-rewriting, $\mathcal{R}^{\mathsf{ir}} = \{Q(x) \leftarrow C(x)$, $Q(x) \leftarrow A(x) \wedge K(x) \wedge \neg B(x), Q(x) \leftarrow A(x) \wedge B(x) \wedge \neg B(x)\}$. When we evaluate $Q(x) \leftarrow C(x)$ of $\mathcal{R}^{\mathsf{ir}}$ over $\mathcal{A}_s$ we yield $\{a\}$ which is not an ICAR-answer. Notice that the saturated ABox $\mathcal{A}_s$ is an upper approximation of $clc(\mathcal{T}, \mathcal{A})$. ♢

ICAR-answering over DL-Lite is FO-rewritable, and therefore in AC^0 in data complexity. However, it was shown that for more expressive DLs, consistent query answering under the ICAR is no longer tractable; actually, it is already coNP-hard in data complexity in $\mathcal{EL}_{\bot nr}$ [18]. Identifying DLs for which ICAR-answering is tractable is quite challenging. It was shown by Rosati [18] that tractability of IAR-answering does not imply tractability of ICAR-answering and the reason is the need to compute *clc*. Despite the theoretical studies over the ICAR semantics [15, 18], there are no algorithms for ICAR-answering over expressive DLs. In the following example we attempt to employ the rewriting approach presented in [15] for an input TBox expressed in $\mathcal{EL}_\bot$.

Example 3. Consider the TBox $\mathcal{T}$ and query $\mathcal{Q}$ of Example 2. In the first step, we compute the IAR-rewriting of $\mathcal{Q}, \mathcal{T}$. For this purpose, we apply the IAR-rewriting algorithm presented in [21] that takes as input an arbitary DL TBox.

$$\begin{aligned}\mathcal{R}^{\mathsf{ir}} = \{\ & Q(x) \leftarrow C(x) && (1)\\ & Q(x) \leftarrow A(x) \wedge K(x) \wedge \neg(A(x) \wedge B(x)) && (2)\\ & Q(x) \leftarrow A(x) \wedge B(x) \wedge \neg(A(x) \wedge B(x))\} && (3)\end{aligned}$$

Next, by following the same approach as in [15], we apply a rewriting procedure on the elements of $\mathcal{R}^{\mathsf{ir}}$ ingoring their negative part (we omit clause (3)):

$$(1) \rightsquigarrow \quad Q(x) \leftarrow A(x) \wedge K(x) \tag{4}$$
$$Q(x) \leftarrow A(x) \wedge B(x) \tag{5}$$
$$(2) \rightsquigarrow Q(x) \leftarrow A(x) \wedge B(x) \wedge \neg(A(x) \wedge B(x)) \tag{6}$$

Finally, we construct the set $\mathcal{R}' = \mathcal{R}^{\mathsf{ir}} \cup \{(4), (5)\}$.

Notice that when $Q(x) \leftarrow A(x) \wedge B(x)$ of $\mathcal{R}'$ is evaluated over $\mathcal{A}$ we obtain $Q(a)$, but $\{a\}$ is not in $\mathsf{cert}_{\mathsf{icar}}(\mathcal{Q}, \mathcal{T}, \mathcal{A})$; hence $\mathcal{R}'$ is not an ICAR-rewriting.

In order to fix this issue, one could check if the clause $\bot \leftarrow A(x) \wedge B(x)$ is entailed from $\mathcal{T}$ to decide whether $Q(x) \leftarrow A(x) \wedge B(x)$ is included in the ICAR-rewriting. In particular, since $\mathcal{T} \models \bot \leftarrow A(x) \wedge B(x)$, any set of the form $\{A(a), B(a)\}$ is inconsistent w.r.t. $\mathcal{T}$, and hence it cannot be used to infer an assertion included in $clc(\mathcal{T}, \mathcal{A})$. Consequently, the clause $Q(x) \leftarrow A(x) \wedge B(x)$ that bounds to assertions of the form $\{A(a), B(a)\}$ cannot be used to yield an ICAR-answer. In the same spirit, the clause $Q(x) \leftarrow A(x) \wedge K(x)$ should be included in the output ICAR-rewriting since it holds that $\mathcal{T} \not\models \bot \leftarrow A(x) \wedge K(x)$. By eliminating (5) from $\mathcal{R}'$ we obtain the ICAR-rewriting $\mathcal{R}^{\mathsf{icr}} = \mathcal{R}^{\mathsf{ir}} \cup \{(4)\}$. $\diamondsuit$

Example 4. Consider the following TBox $\mathcal{T}$, query $Q(x) \leftarrow A(x)$ and ABox $\mathcal{A} = \{R(a, b), K(b), R(b, a)\}$.

$$\mathcal{T} = \{\ A(x) \leftarrow R(x, y) \wedge K(y) \tag{7}$$
$$A(x) \leftarrow R(x, y) \wedge A(y) \tag{8}$$
$$\bot \leftarrow K(x) \wedge R(x, y)\} \tag{9}$$

We first compute the IAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$ by applying the calculus of [21]:

$$\mathcal{R}^{\mathsf{ir}} = \{\ Q(x) \leftarrow A(x) \tag{10}$$
$$A(x) \leftarrow R(x, y) \wedge K(y) \wedge \neg(R(x, y) \wedge K(x)) \wedge \neg(R(y, z) \wedge K(y)) \tag{11}$$
$$A(x) \leftarrow R(x, y) \wedge A(y) \wedge \neg(R(x, y) \wedge K(x))\} \tag{12}$$

Next, we apply a rewriting procedure on the elements of $\mathcal{R}^{\mathsf{ir}}$:

$$(10), (12) \rightsquigarrow A(x) \leftarrow R(x, y) \wedge K(y) \qquad (7)$$
$$A(x) \leftarrow R(x, y) \wedge A(y) \qquad (8)$$

In line with the Example 4 notice that for the clauses (7),(8) in $\mathcal{R}'$ it holds that $\mathcal{T} \not\models \bot \leftarrow R(x, y) \wedge K(y)$, $\mathcal{T} \not\models \bot \leftarrow R(x, y) \wedge A(y)$. However, $\mathcal{R}' = \mathcal{R}^{\mathsf{ir}} \cup \{(7), (8)\}$ is not an ICAR-rewriting. Indeed, when we evaluate (7) over $\mathcal{A}$ we obtain $A(a)$ and because of (8) we derive $A(b)$. However, b is not an ICAR-answer since $A(b) \notin clc(\mathcal{T}, \mathcal{A})$. This is because to derive $A(b)$ we have used $\{R(a, b), K(b), R(b, a)\}$ which is inconsistent w.r.t. $\mathcal{T}$. $\diamondsuit$

As illustrated in Example 4 in order to introduce a recursive clause of the form $A(x) \leftarrow R(x,y) \wedge A(y)$ in the ICAR-rewriting it is not sufficient to examine if $\mathcal{T} \not\models \bot \leftarrow R(x,y) \wedge A(y)$; since the concept A participates in a recursion, there is an infinite number of negative clauses for which we should examine if they are entailed from $\mathcal{T}$. Intuitively, this is the reason for the co-NP data complexity [18] of the ICAR-answering problem: if the input query contains concepts involved in some recursion (such as concept A in our example), then the number of ABox assertions that can be used to infer an assertion in $clc(\mathcal{T}, \mathcal{A})$ is unbounded.

4 ICAR-rewriting over expressive DLs

Based on the ideas presented in Section 3 we propose an algorithm for ICAR-rewriting over a TBox expressed in an arbitary DL. Definition 5 describes the notion of the negative closure that was used in [21] to obtain the IAR-rewriting. Intuitively, the negative closure $\mathcal{T}_{cn}$ of a TBox $\mathcal{T}$ is a finite set of negative clauses that can capture the negative clauses entailed from $\mathcal{T}$. We use $\mathcal{T}_{cn}$ to examine if the condition described in Example 3 holds.

Definition 5. *A* negative closure *of a 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 $\bot \leftarrow \bigwedge \alpha_i \models \bot \leftarrow \bigwedge \beta_i$.*

Algorithm 1 ICAR-Rewriting

```
   Input: a CQ Q and a L-TBox T
 1: Compute a negative closure T_cn of T
 2: Compute the IAR-rewriting R^ir of Q w.r.t. T.
 3: R^icr := R^ir
 4: for H(s) ← ⋀_i α_i ∧ ⋀_j ¬β_j ∈ R^ir  do
 5:     Compute a UCQ-rewriting R_α of Q(s) ← ⋀_i α_i w.r.t. T
 6:     for each Q(s) ← ⋀_i α'_i ∈ R_α do
 7:         if for every clause C ∈ T_cn it holds C ⊭ ⊥ ← ⋀_i α'_i then
 8:             R^icr = R^icr ∪ {H(s) ← ⋀_i α'_i ∧ ⋀_j ¬β_j}
 9:         end if
10:     end for
11: end for
12: return R^icr
```

Algorithm 1 computes the IAR-rewriting $\mathcal{R}^{\mathsf{ir}}$ (line (2)) by applying the procedure presented in [21]. Then, in order to build the set $\mathcal{R}^{\mathsf{icr}}$, it applies a rewriting procedure on the elements of $\mathcal{R}^{\mathsf{ir}}$ by neglecting their negative part. More precisely, for the positive body atoms α_i of every element in $\mathcal{R}^{\mathsf{ir}}$, it constructs the UCQ-rewriting of the query $Q(\vec{s}) \leftarrow \bigwedge_i \alpha_i$ (lines (4)-(5)). Condition in line (7) is necessary, as explained in Section 3, in order to only retrieve facts from $clc(\mathcal{T}, \mathcal{A})$ when evaluating $\mathcal{R}^{\mathsf{icr}}$ over $\mathcal{A}$.

Example 5. Consider the following $\mathcal{EL}_\bot$ TBox $\mathcal{T}$ and query $Q(x) \leftarrow A(x)$.

$$\mathcal{T} = \{A(x) \leftarrow R(x,y) \wedge B(y) \tag{13}$$
$$B(x) \leftarrow C(x) \tag{14}$$
$$\bot \leftarrow B(x) \wedge K(x)\} \tag{15}$$

In the first step, Algorithm 1 computes the IAR-rewriting of $\mathcal{Q}$:

$$\mathcal{R}^{\mathsf{ir}} = \{\ Q(x) \leftarrow A(x) \tag{16}$$
$$A(x) \leftarrow R(x,y) \wedge B(y) \wedge \neg(B(y) \wedge K(y)) \tag{17}$$
$$A(x) \leftarrow R(x,y) \wedge C(y) \wedge \neg(C(y) \wedge K(y))\} \tag{18}$$

Next, by considering the clauses (16), (17), (18) in $\mathcal{R}^{\mathsf{ir}}$ it computes the UCQ-rewriting of $Q(x) \leftarrow A(x)$, $Q(x) \leftarrow R(x,y) \wedge B(y)$, $Q(x) \leftarrow R(x,y) \wedge C(y)$:

$$(16) \rightsquigarrow Q(x) \leftarrow R(x,y) \wedge B(y) \tag{19}$$
$$Q(x) \leftarrow R(x,y) \wedge C(y) \tag{20}$$
$$(17) \rightsquigarrow A(x) \leftarrow R(x,y) \wedge C(y) \wedge \neg(B(y) \wedge K(y)) \tag{21}$$

The negative closure of $\mathcal{T}$ is $\mathcal{T}_{cn} = \{\bot \leftarrow B(x) \wedge K(x), \bot \leftarrow C(x) \wedge K(x)\}$. Therefore, the clauses $\bot \leftarrow R(x,y) \wedge B(y)$, and $\bot \leftarrow R(x,y) \wedge C(y)$ are not entailed by $\mathcal{T}_{cn}$ and condition in line (7) is satisfied. Finally, Algorithm 1 outputs $\mathcal{R}^{\mathsf{icr}} = \mathcal{R}^{\mathsf{ir}} \cup \{(19), (20), (21)\}$ that is an ICAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$. $\diamondsuit$

Theorem 1. *Let $\mathcal{T}$ be a $\mathcal{L}$ TBox where $\mathcal{L}$ is a Horn-DL, and let $\mathcal{Q}$ be a CQ. If $\mathcal{Q}$ is UCQ-rewritable w.r.t. $\mathcal{T}$ and there exists a negative closure $\mathcal{T}_{cn}$ of $\mathcal{T}$, then Algorithm 1 terminates and computes the ICAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$.*

Proof. (sketch) In [21] it was shown that if there exists a negative closure of $\mathcal{T}$, and $\mathcal{Q}$ is datalog rewritable, then there always exists an IAR-rewriting $\mathcal{R}^{\mathsf{ir}}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$ (Theorem 6). Therefore, if there exists a negative closure of $\mathcal{T}$ and a UCQ-rewriting $\mathcal{R}$ of $\mathcal{Q}$ w.r.t. $\mathcal{T}$, then there also exists an IAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$, and Algorithm 1 terminates. To prove correctness of Algorithm 1 we first show that $\mathcal{R}^{\mathsf{icr}} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$ iff $\mathcal{R}^{\mathsf{ir}} \cup clc(\mathcal{A}) \models \mathcal{Q}(\vec{a})$. By definition of $\mathcal{R}^{\mathsf{ir}}$ it holds that $\mathcal{R}^{\mathsf{ir}} \cup clc(\mathcal{A}) \models \mathcal{Q}(\vec{a})$ iff $\mathcal{R} \cup clc(\mathcal{A}) \models_{\mathsf{ir}} \mathcal{Q}(\vec{a})$. Finally, by definition of $\mathcal{A}_{\mathsf{icar}}$ we conclude that $\mathcal{R}^{\mathsf{icr}} \cup \mathcal{A} \models \mathcal{Q}(\vec{a})$ iff $\mathcal{R} \cup \mathcal{A} \models_{\mathsf{icr}} \mathcal{Q}(\vec{a})$.

□

5 Positive Results for ICAR-answering

In this section we exploit results on UCQ-rewritability of queries over a range of DLs and the ICAR-rewriting approach presented in Section 4, to provide positive results for ICAR-answering over Horn-DLs that do not fall into the DL-Lite fragment.

The authors in [5] showed that instance queries over semi-acyclic-$\mathcal{EL}_\bot$ TBoxes are always UCQ-rewritable. Moreover, in [21] it was shown that there always exists a negative closure for a semi-acyclic-$\mathcal{EL}_\bot$ TBox. Theorem 2 follows.

Theorem 2. *Let $\mathcal{T}$ be a semi-acyclic-$\mathcal{EL}_\bot$ TBox and let $\mathcal{Q}$ be an instance query. Then, on input $\mathcal{T}$ and $\mathcal{Q}$, Algorithm 1 terminates and computes an ICAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$.*

In [18] the $\mathcal{EL}_{\bot nr}$ fragment of $\mathcal{EL}_\bot$ was studied and in [21] it was shown that there always exists a negative closure of an $\mathcal{EL}_{\bot nr}$ TBox. Therefore, although the problem of ICAR-answering of CQs over $\mathcal{EL}_{\bot nr}$ TBoxes is in general intractable, for a given UCQ-rewritable CQ we obtain the following result.

Theorem 3. *Let $\mathcal{T}$ be a $\mathcal{EL}_{\bot nr}$ TBox and let $\mathcal{Q}$ be a CQ that is UCQ-rewritable. Then, on input $\mathcal{T}$ and $\mathcal{Q}$ Algorithm 1 terminates and computes an ICAR-rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{T}$.*

To check if a given query is UCQ-rewritable we can exploit results in UCQ-rewritability of queries over DLs that are not always UCQ-rewritable [6, 4, 11]. The authors in [6] study UCQ-rewritability of a given instance query over Horn-DLs, like $\mathcal{EL}_\bot$, $\mathcal{ELI}_\bot$ and Horn-$\mathcal{SHIF}$. These results were used to design a practical algorithm for checking UCQ-rewritability of instance queries [11]. Subsequently, the system of [11] was extended to support rooted CQs [10].

Moreover, one can also check if a negative closure $\mathcal{T}_{cn}$ exists for a given TBox, $\mathcal{T}$, by using the condition presented in [21]. This condition is described in Lemma 1. Intuitively, the non-existence of $\mathcal{T}_{cn}$ is related to concepts in the negative clauses of $\mathcal{T}$ that participate in some recursion.

Lemma 1. *Let $\mathcal{T}$ be a $\mathcal{L}$ TBox where $\mathcal{L}$ is a Horn-DL. Let the set of 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 there exists a negative closure T_{cn} of $\mathcal{T}$.*

Therefore, given an TBox $\mathcal{T}$ expressed in a Horn-DL one can decide on the existence of a negative closure by using the system of [11] to check UCQ-rewritability of all relevant instance queries described in Lemma 1. Moreover, the system of [11] can be used to check UCQ-rewritability of the input query $\mathcal{Q}$. If the conditions of Theorem 1 are satisfied, Algorithm 1 can be used to obtain an ICAR-rewriting of $\mathcal{Q}, \mathcal{T}$.

6 Evaluation

We have created a prototype system to perform a preliminary experimental evaluation of the proposed framework. Our system is based on the implementation of Algorithm 1. At first step, the UCQ-rewritability of the input query $\mathcal{Q}$ is examined by using the system Grind [10]. Next, our system uses the IAR-rewriting framework implemented in [21] to decide if there exists a negative closure of the input TBox $\mathcal{T}$ and if so, to compute an IAR-rewriting of $\mathcal{Q}$ and $\mathcal{T}$. If $\mathcal{Q}$ is not UCQ-rewritable, or if a negative closure cannot be computed for $\mathcal{T}$, then the system reports that it cannot output an ICAR-rewriting. Otherwise, it proceeds

	$t_{\mathcal{R}}$	$\lvert\mathcal{R}^{\mathsf{icr}}\rvert$	$q^{\neg}$	max	avg		$t_{\mathcal{R}}$	$\lvert\mathcal{R}^{\mathsf{ir}}\rvert$	$q^{\neg}$	max	avg
envo	1 889	44 623	96%	7	6.5	MOHSE	17 037	98 119	96%	50	1
	1 025	21 171	96%	7	6.4		32 213	101 646	96%	50	1
	1 140	21 170	96%	7	6.5		15 651	101 620	96%	2	1.1
	1 091	21 927	96%	14	6.7		1 049	3 511	92%	2	1.1
	1314	22 694	96%	7	6.5		1 701	31	80%	50	50
	129	75	86%	4	4.0		1 923	3	66%	2	2.0
	1 441	21 932	97%	7	6.5		1 269	43	72%	50	50.0
	1 126	21 170	96%	7	6.5		1 314	43	51%	50	50
	2 538	21 170	96%	7	6.5		2 9375	98115	96%	50	21.5
	2 385	21 934	96%	7	6.5		34 318	98115	96%	50	21
FBbi	194	285	7%	7	4.2	Not-Galen	178 351	82 885	80%	6	1.0
	148	406	5%	7	3.2		175 581	82 885	80%	6	1.6
	85	13	100%	13	4.1		172 970	82 886	80%	6	1.57
	88	29	100%	4	2.8		36	33	36%	6	1.7
	51	307	7%	7	4.2		176 622	83 884	80%	6	1.0
	72	9	100%	10	2.8		176 102	82 886	80%	6	1.0
	90	555	54%	7	3.5		171 392	82 885	80%	5	1.0
	76	295	51%	1	1.0		32	41	51%	1	1.0
	56	280	8%	7	4.2		132 558	82 885	80%	6	1.0
	90	547	53%	7	3.9		170 489	82 885	80%	6	1.0

Table 2: Results for computation of IAR-rewritings.

in computing the ICAR-rewriting $\mathcal{R}^{\mathsf{icr}}$ as described in lines 3-10. The whole system currently supports ontologies expressed in $\mathcal{EL}_{\bot}$ which is the DL supported by the current implementation of Grind.

To generate our experimental setting we examined the ontologies ENVO, FBbi, MOHSE, NBO, Not-Galen that were used in [10] to evaluate Grind. We did not consider SO as it was reported in [21] that a negative closure cannot be constructed for this ontology. The ontologies ENVO and FBbi include negative axioms. For the rest ontologies, that is MOHSE and Not-Galen, we manually added negative axioms. For this purpose we tried to use concepts that appear in different levels in the concepts hierarchy, so that these affect large or small parts of the ontology. Each ontology used in [10] came with 10 handcrafted queries. Among them we used only those queries that include concepts involved in some negative axiom and for which the Grind system reported they are UCQ-rewritable. We have also manually constructed test queries that each one of them contains at least one body atom that uses a concept or role involved in a negative axiom. More precisely, for an axiom 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 $\mathcal{T} \models B \sqsubseteq D$. Overall, for each ontology we used 10 test queries that satisfy the UCQ-rewritability condition.

Our results are depicted in Table 2. Columns $t_{\mathcal{R}}$, $\lvert\mathcal{R}^{\mathsf{ir}}\rvert$ and $q^{\neg}$ present the time to obtain the ICAR-rewriting in ms, the number of clauses in the output

rewriting, and the percentage of the clauses in the output that contain a negative part. Finally columns max and avg present the maximum and average number of negative atoms in the elements of the rewriting. In most cases the ICAR-rewriting was obtained within a few seconds. In contrast, in the case of Not-Galen the time to compute the rewriting was up to 3 minutes. This is because, the rewriting procedure (described in line 5, Algorithm 1) was applied on every element of the IAR-rewriting which was quite large for almost all test queries. One could avoid the several calls of the rewriting procedure and exploit the rewritings that have already been constructed during the IAR-rewriting process. For example, for an input TBox $\mathcal{T} = \{A(x) \leftarrow A_1(x), \bot \leftarrow A(x) \wedge B(x)\}$ and query $\mathcal{Q} = Q(x) \leftarrow A(x) \wedge C(x)$ the IAR-rewriting is of the form $\mathcal{R}^{\mathsf{ir}} = \{Q(x) \leftarrow A(x) \wedge C(x) \wedge \neg(A(x) \wedge B(x)) \wedge \neg(A_1(x) \wedge B(x)), Q(x) \leftarrow A_1(x) \wedge C(x) \wedge \neg(A_1(x) \wedge B(x))\}$ and to obtain $\mathcal{R}^{\mathsf{ir}}$ the standard rewriting $\mathcal{R} = \{Q(x) \leftarrow A(x) \wedge C(x), Q(x) \leftarrow A_1(x) \wedge C(x)\}$ must be computed. Therefore, to construct the ICAR-rewriting one could make use of $\mathcal{R}$ instead of applying anew rewriting procedure on every element of $\mathcal{R}^{\mathsf{ir}}$. Our implementation does not involve such optimisations however we feel that they could reduce the rewriting times.

Regarding the size of the output rewriting, one could design optimisations to eliminate the redudant elements from the output, $\mathcal{R}^{\mathsf{icr}}$. For example, the clause $Q(x) \leftarrow A_1(x) \wedge C(x) \wedge \neg(A(x) \wedge B(x)) \wedge \neg(A_1(x) \wedge B(x))$ is subsumed by $Q(x) \leftarrow A_1(x) \wedge C(x) \wedge \neg(A_1(x) \wedge B(x))$ and hence the former can be discarded from $\mathcal{R}^{\mathsf{icr}}$. Further work is required in that respect to reduce the size of $\mathcal{R}^{\mathsf{icr}}$.

Finally, the number of negative conjuncts in the elements of the rewriting was quite small (up to 50). Note that the evaluation in [22] showed that triple-store systems can handle a large number of negative atoms (even more than one hundred). In conclusion, in most cases we have been able to obtain within reasonable time an ICAR-rewriting for our test ontologies and queries.

Summarizing, our evaluation results show that we were able, in most cases, to compute an ICAR-rewriting for the given TBoxes for which the problem is in general intractable. Moreover, computing the ICAR-rewriting can be done relatively efficiently and the number of negative atoms added in the clauses was usually quite small.

7 Conclusions

In this work we have provided a general framework for ICAR-answering for arbitary Horn-DLs. We have presented an algorithm that takes as an input a TBox and a query; if it terminates, it outputs a datalog program, that can be used to compute the ICAR-answers. We described the termination condition of our algorithm and showed that the tractability results hold for the semi-acyclic-$\mathcal{EL}_\bot$. Furthermore, in cases of inputs that do not satisfy the termination condition in general, we can exploit recent results to check termination. Our experiments provided encouraging results as in almost all cases we were able to compute an ICAR-rewriting in reasonable time. Further experimental evaluation to examine whether the conditions apply in practice is left for future work.

References

1. Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite Family and Relations. *Journal of Artificial Intelligence Research*, 36:1–69, 2009.
2. Franz Baader, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2002.
3. Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué. Querying Inconsistent Description Logic Knowledge Bases under Preferred Repair Semantics. In *AAAI*, pages 996–1002, 2014.
4. Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics. In *IJCAI: International Joint Conference on Artificial Intelligence*, 2016.
5. Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. Deciding FO-Rewritability in $\mathcal{EL}$. In *Proceedings of the Twenty-Fifth International Workshop on Description Logics*, 2012.
6. Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. First-Order Rewritability of Atomic Queries in Horn Description Logics. In *Proceeding of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.
7. Meghyn Bienvenu and Riccardo Rosati. New Inconsistency-Tolerant Semantics for Robust Ontology-Based Data Access. In *Proceedings of the Twenty-Sixth International Workshop on Description Logics*, 2013.
8. Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. *ACM Transactions on Database Systems*, 39(4):33:1–33:44, 2014.
9. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.
10. Peter Hansen and Carsten Lutz. Computing FO-Rewritings in *EL* in Practice: From Atomic to Conjunctive Queries. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, pages 347–363, 2017.
11. Peter Hansen, Carsten Lutz, Inanç Seylan, and Frank Wolter. Efficient Query Rewriting in the Description Logic $\mathcal{EL}$ and Beyond. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3034–3040, 2015.
12. Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev. Conjunctive Query Answering with OWL 2 QL. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2012.
13. Atanas Kiryakov, Barry Bishoa, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. The features of bigowlim that enabled the bbcs world cup website. In *Workshop on Semantic Data Management (SemData)*, pages 13–17, 2010.
14. Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant Semantics for Description Logics. In *Proceedings of Fourth International Conference on Web Reasoning and Rule Systems*, pages 103–117. Springer, 2010.

15. Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Query rewriting for inconsistent DL-Lite ontologies. In *Proceedings of the Fifth International Conference on Web Reasoning and Rule Systems*, pages 155–169. Springer, 2011.
16. Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant Query Answering in Ontology-Based Data Access. *Journal of Web Semantics*, 33:3–29, 2015.
17. Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. Tractable Query Answering and Rewriting under Description Logic Constraints. *Journal of Applied Logic*, 8(2):186–209, 2010.
18. Riccardo Rosati. On the Complexity of Dealing with Inconsistency in Description Logic Ontologies. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 1057–1062, 2011.
19. Giorgos Stoilos, Bernardo Cuenca Grau, Boris Motik, and Ian Horrocks. Repairing Ontologies for Incomplete Reasoners. In *Proceedings of the 10th International Semantic Web Conference, Bonn, Germany*, pages 681–696, 2011.
20. Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos Stamou. Optimising Resolution-Based Rewriting Algorithms for OWL Ontologies. *Journal of Web Semantics*, 33:30–49, 2015.
21. Despoina Trivela, Giorgos Stoilos, and Vasilis Vassalos. A Framework and Positive Results for IAR-answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.
22. Eleni Tsalapati, Giorgos Stoilos, Giorgos B. Stamou, and George Koletsos. Efficient Query Answering over Expressive Inconsistent Description Logics. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1279–1285, 2016.