Efficient Query Answering Over Expressive Inconsistent Description Logics

Eleni Tsalapati, 1 Giorgos Stoilos, 2 Giorgos Stamou, 1 and George Koletsos 1

1 School of Electrical and Computer Engineering, National Technical University of Athens, Greece
2 Department of Informatics, Athens University of Economics and Business

Abstract

Inconsistent-tolerant semantics, like the IAR and ICAR semantics, have been proposed as means to compute meaningful query answers over inconsistent Description Logic (DL) ontologies. In the current paper we present a framework for scalable query answering under both the IAR and ICAR semantics, which is based on highly efficient data saturation systems. Our approach is sound and complete for ontologies expressed in the lightweight DL DL-Lite, but for more expressive DLs the problem is known to be intractable, hence our algorithm only computes upper approximations. Nevertheless, its structure motivates a new type of ICAR-like semantics which can be computed in polynomial time for a very large family of DLs. We have implemented our techniques and conducted an experimental evaluation obtaining encouraging results as both our IAR- and ICAR-answering approaches are far more efficient than existing available IAR-based answering systems.

1 Introduction

Conjunctive query answering over data described using ontological knowledge is a key reasoning service for many modern applications [Motik et al., 2012; Chaussecurte et al., 2013]. Although query answering is normally defined over consistent datasets, in real-world applications the data can very often be inconsistent with respect to the axioms specified in the ontology. In this case the straightforward approach would be to try and resolve the inconsistencies by “cleaning” the dataset from the conflicting elements. However, in many occasions this might not be practical due to, e.g., the large volume of the data and the non-deterministic nature of the cleaning procedures. A second approach that has been proposed in the literature, called consistent query answering, is to devise semantics which describe which answers are “meaningful” to be returned even in the presence of the inconsistencies [Arenas et al., 1999; Bertossi, 2006]. For this reason such semantics are referred to as inconsistent-tolerant.

Consistent query answering has recently been studied in the field of Description Logics (DLs) [Lembo et al., 2010; Rosati, 2011; Bienvenu and Rosati, 2013; Bienvenu et al., 2014]. Two important approaches, mostly due to their nice computational properties over the lightweight DL DL-Lite, consist of the so-called IAR and ICAR semantics [Lembo et al., 2010]. Despite important theoretical results and a few implemented systems [Masotti et al., 2011; Rosati et al., 2012; Bienvenu et al., 2014], designing practically efficient consistent query answering systems that could scale up to billions of data is still largely open. Especially for the ICAR semantics, only a preliminary effort was reported by Masotti et al. [2011] but the evaluation used very small proprietary datasets (of the scale of a few thousands). Furthermore, over many still lightweight DLs, like EL, ICAR-based query answering is intractable [Rosati, 2011] and, to the best of our knowledge, no approximate algorithms exist in the literature; Bienvenu and Rosati [2013] provided approximations of a different type of inconsistent-tolerant semantics called AR semantics.

In this work we present a framework for efficient query answering under both the IAR and ICAR semantics. First, we focus on the ICAR semantics and provide a query answering framework that is based on highly efficient mature data saturation (triple-store) systems. This is particularly interesting as these systems have shown to be able to handle billions of data. Moreover, their properties enable us to propose additional refinements and optimisations. Our algorithm is correct (sound and complete) for ontologies expressed in languages of the DL-Lite family. However, as mentioned, ICAR-based query answering over DLs like EL is intractable [Rosati, 2011], hence our algorithm only computes upper approximations of them. Nevertheless, its structure motivates a new type of ICAR-like semantics which we show that can be computed in polynomial time for a very large number of highly expressive DLs; none such semantics were previously known. Subsequently, we show that our framework can also be used for IAR-based query answering for ontologies expressed in the DLs DL-Lite and $\mathcal{EL}_{\mathit{int}}$ [Rosati, 2011]. Finally, we have conducted an experimental evaluation obtaining encouraging results as both our approaches (IAR and ICAR) are more efficient than existing IAR-answering systems [Rosati et al., 2012; Bienvenu et al., 2014].

*At the time of acceptance Giorgos Stoilos was a member of NTUA supported by a Marie Curie Career Reintegration Grant within European Union’s 7th Framework Programme (FP7/2007-2013) under REA grant agreement 303914.
2 Preliminaries

We use standard notions of first-order constants, variables, atoms, satisfiability, entailment (|=), model, substitutions (σ), etc. We also assume familiarity with datalog. For a conjunction of atoms \( B = \alpha_1 \land \ldots \land \alpha_n \) we often abuse notation and write \( B \subseteq S \) to denote that \( \{\alpha_1, \ldots, \alpha_n\} \subseteq S \).

2.1 Description Logics

Description Logics (DLs) [Baader et al., 2003] constitute a family of (mostly) decidable fragments of FOL. We use \( L \) to denote an arbitrary DL language and next we recapitulate the syntax of the DLs \( EL \) [Baader et al., 2005], DL-Lite [Calvanese et al., 2007], and RL [Motik et al., 2009].

The set of \( EL \)-concepts is inductively defined by the grammar: \( C : = \bot \mid A \mid C_1 \sqcap C_2 \mid \exists R.C \), where \( A \) (\( R \)) is called atomic concept (role) and \( C_{(i)} \) are \( EL \)-concepts. An \( EL \)-ABox \( \mathcal{T} \) is a finite set of inclusions of the form \( C_1 \sqsubseteq C_2 \), or \( \exists R.C \). Inclusions of the form \( C_1 \sqcap C_2 \subseteq \bot \) are called negative and the rest positive. We will also refer to \( EL \)-ABox-saturation, a syntactic restriction of \( EL \) defined by Rosati [2011].

DL-Lite restricts \( EL \) by allowing only for concepts of the form \( A \) or \( \exists H.T \); however, \( R \) in DL-Lite can also be the inverse of a role of the form \( \neg R \) and we can also have inclusions of the form \( S \sqsubseteq R \) or \( S \sqsubseteq \neg R \) for \( S, R \) roles. Finally, roughly speaking, \( RL \) is defined as those DL axioms that can be translated (by simple syntactic transformations) into datalog. For example, \( \exists R.A \sqsubseteq B \) is an \( RL \) axiom since it corresponds to the datalog rule \( R(x, y) \land (\neg R(y)) 

2.2 Queries and Query Answering

A conjunctive query (CQ) is an expression of the form \( \exists j.\phi(x, y) \), where \( \phi \) is a conjunction of function-free atoms containing only variables from \( x \) (called answer variables) or from \( y \) (called existential variables). A set of conjunctive queries (UCQ) is a set of CQs all having the same number of answer variables. A tuple of constants \( \alpha \) is a certain answer of a query \( Q \) over \( T \cup A \) if \( T \cup A \models \exists j.\phi(\alpha, y) \). We denote with \( \text{cert}(Q, T \cup A) \) all the certain answers of \( Q \) over \( T \cup A \).

Query answering can be realised via a technique known as query rewriting [Calvanese et al., 2007] which we recall next.

Definition 1. Let \( Q \) be a CQ and let \( T \) be a TBox. A datalog rewriting (or simply rewriting) \( \mathcal{R} \) for \( Q \), \( T \) is a pair \((\mathcal{R}_Q, \mathcal{R}_D)\), where \( \mathcal{R}_Q \) is a UCQ, \( \mathcal{R}_D \) is a datalog program, and for each ABox \( A \) consistent w.r.t. \( T \) we have:

\[ \text{cert}(Q, T \cup A) = \text{cert}(\mathcal{R}_Q, \mathcal{R}_D \cup A) \]

If \( \mathcal{R}_D = \emptyset \) then \( \mathcal{R} \) is called UCQ rewriting and instead of \((\mathcal{R}_Q, \emptyset)\) we simply use \( \mathcal{R}_Q \) to identify the rewriting.

A DL \( L \) is called datalog rewritable if for every \( L \)-TBox \( T \) a datalog program \( \mathcal{R}_D \) exists such that for every query \( Q \) without existential variables \( \mathcal{R} = (\{Q\}, \mathcal{R}_D) \) is a datalog rewriting for \( Q, T \).

2.3 RL query answering systems

Since \( RL \)-axioms correspond to datalog rules, to perform query answering systems that support this DL follow a datalog-like saturation approach based on the \( RL \) fragment of some TBox \( T \), which in the following we denote by \( T \upharpoonright \).

Definition 2. An \( RL \) ABox-saturation system ans is a procedure that takes as input a TBox \( T \), an ABox \( A \), and a CQ \( Q \) and returns a set of tuples \( \text{ans}(Q, T \cup A) = \text{cert}(Q, A) \), where \( A \subseteq A \), called saturation, contains all assertions \( \alpha \) such that \( T \upharpoonright \cup A \models \alpha \).

Most systems known to us, like GraphDB, Oracle’s Semantic Graph, and RDFox are \( RL \) ABox-saturation systems.

2.4 Inconsistency-tolerant Query Answering

In order to return meaningful answers even from inconsistent datasets, consistent query answering has been introduced in the areas of databases and ontologies. Next, we recapitulate the so-called IAR and ICAR semantics [Lembo et al., 2011].

Definition 3. For a TBox \( T \) and an ABox \( A \) let \( \text{clc}(T, A) = \{\alpha \mid \text{some consistent subset } S \subseteq A \exists s.t. T \upharpoonright \cup S \models \alpha \} \). The Intersection Closed ABox Repair (ICAR) of \( T \cup A \) is defined to be the intersection of all subsets \( A' \subseteq \text{clc}(T, A) \) that are consistent w.r.t. \( T \) and no other \( A'' \) consistent w.r.t. \( T \) exists s.t. \( A' \subseteq A'' \subseteq \text{clc}(T, A) \).

Let \( Q \) be a CQ and let \( A^\mathcal{E} \) be the ICAR of \( T \cup A \). A tuple \( \alpha \) is an ICAR-answer of \( Q \) w.r.t. \( T \cup A \) if \( \alpha \in \text{cert}(Q, T \cup A^\mathcal{E}) \). We denote by \( \text{cert}^\mathcal{E}(Q, T \cup A) \) all ICAR-answers of \( Q \) w.r.t. \( T \cup A \). The IAR semantics and answers (denoted by \( \text{cert}^\mathcal{I}(Q, T \cup A) \)) are defined similarly by replacing \( \text{clc}(T, A) \) with \( A \).

An algorithm to compute IAR/ICAR-answers over DL-Lite-TBoxes was presented by Lembo et al. [2011]: Let \( R \) be a UCQ rewriting for some query \( Q \) and TBox \( T \). Evaluating \( R \) over \( A \) would return spurious IAR/ICAR-answers; hence each \( Q \in \mathcal{K} \) needs to be extended with proper negative atoms which will guarantee that only answers that are supported by the intersection of all repairs are returned. Roughly speaking, for each atom \( A(x) \) that appears in \( Q \) if \( T \models A(x) \) then \(-B(x)\) is added to \( Q \). We denote this procedure by \( \text{ref}(R, T) \) and illustrate it with an example; for details please see [Lembo et al., 2011, Section 7.2.2].

Example 4. Consider the following TBox and ABox:

\[ T = \{\exists P \sqsubseteq B, P \sqsubseteq \neg P', B \sqsubseteq \neg B'\}, \]
\[ A = \{P(a, b), P'(a, b), P'(c, d), B(e), B'(e)\} \]

as well as query \( Q = B(x) \). Clearly, \( A \) is inconsistent w.r.t. \( T \) (e.g., due to \( \{B(e), B'(e), B \sqsubseteq \neg B'\} \), however, we have \( \text{cert}^\mathcal{I}(Q, T \cup A) = \{e\} \) and \( \text{cert}^\mathcal{E}(Q, T \cup A) = \{a, c\} \) (“\( a \)” is an ICAR-answer since \( B(a) \in \text{clc}(T, A) \)). Now, \( \mathcal{R}_Q = \{B(x), \exists y.P'(x, y)\} \) is a UCQ rewriting for \( Q, T \), then, \( \text{ref}(\mathcal{R}_Q, T) \) returns the following set:

\[ \{B(x) \land \neg B'(x), \exists y.(P'(x, y) \land \neg P'(x, y)) \land \neg B'(x)\} \]

It is easy to see that if we evaluate \( \text{ref}(\mathcal{R}_Q, T) \) on \( A \) we obtain \( \{e\} \), i.e., the IAR-answers; notice how the negative atom
\(-B'(x)\) prevents “\(e\)” from being returned as an answer. Finally, to obtain the ICAR-answers, a rewriting algorithm is applied a second time over \(\text{ref}(R_Q, T)\) returning the set:
\[
\text{ref}(R_Q, T) \cup \{ \exists y, P(x, y) \wedge \neg B'(x) \}
\]
Evaluating the above over \(A\) will indeed return \(\{a, c\}\). \(\diamondsuit\)

3 Efficient ICAR-Answering Over DL-Lite

Interestingly, the saturation computed by ABox-saturation systems given \(T\) and \(A\) most likely consists of an approximation of \(\text{cel}(T, A)\). Hence, these systems could (potentially) be used for efficiently computing ICAR-answers.

**Example 5.** Consider \(T, A,\) and \(Q\) from Example 4 as well as some RL ABox-saturation system \(\text{ans}\). Since \(T^+\) is an RL-TBox, over \(T^+ \cup A\) ans will compute the saturation \(A_s = A \cup \{B(a), B(b)\}\). Moreover, \(\text{ref}(Q, T) = \{B(x) \wedge \neg B'(x)\}\) which evaluated over \(A_s\) returns the set \(\{a, c\}\), i.e., precisely the ICAR-answers. Note that since we want to compute answers in the presence of inconsistencies rather than allow ans to report them, we discarded the negative inclusions and considered how ans behaves over \(T^+ \cup A\).

However, in TBoxes that contain non-RL-axioms such systems will miss certain (ICAR-)answers. To overcome this issue we consider the TBox completion framework introduced by Stoilos et al. [2011; 2014]. Intuitively, in many cases “materialising” certain entailed axioms of the TBox “helps” the RL system compute all certain answers even over non-RL-TBoxes. We recall this notion next.

**Definition 6** ([Stoilos et al., 2011; Stoilos, 2014]). Let \(T\) be a TBox and let \(\text{ans}\) be an RL ABox-saturation system. A completion of \(T\) for \(\text{ans}\) is a set of axioms \(C\) such that \(T \models C\) and for every ABox \(A\) consistent w.r.t. \(T\) and for every CQ \(Q\) without existential variables we have: \(\text{cert}(Q, T \cup A) \subseteq \text{ans}(Q, T \cup C \cup A)\).

Moreover, let \(Q\) be some arbitrary CQ. If a rewriting \(\langle R_Q, R_D \rangle\) for \(Q, T\) exists then we have: \(\text{cert}(Q, T \cup A) \subseteq \text{ans}(R_Q, T \cup C \cup A)\).

**Example 7.** Consider again \(T, A,\) and \(Q\) from Example 4 extended to \(T'\) and \(A'\) as follows:
\[
T' = \{C \subseteq \exists P\} \cup T \quad A' = \{C(f)\} \cup A
\]
Clearly, \(\text{cert}^s(Q, T' \cup A') = \text{cert}^s(Q, T' \cup A) \cup \{f\}\). Consider also an RL ABox-saturation system \(\text{ans}\). Since \(T_{\mid \text{ans}} = \{\exists P \subseteq B\}\) there are clearly ABoxes for which \(\text{ans}\) will miss answers of \(Q\). For example, we have \(f \in \text{cert}(Q, T' \cup \{C(f)\})\) but \(f \notin \text{ans}(Q, T' \cup \{C(f)\})\) since \(T_{\mid \text{ans}} \cup \{C(f)\} \neq B(f)\). Hence, the ICAR-answer “\(f\)” cannot be computed by the procedure illustrated in Example 5.

However, consider the completion \(C = \{C \subseteq B\}\) of \(T\) for \(\text{ans}\). Then, over \(T_{\mid \text{ans}} \cup C \cup A\) system ans will compute the saturation \(A_s = A \cup \{B(b)\}\) since \(T_{\mid \text{ans}} \cup C \cup A \models B(f)\).
Finally, evaluating \(B(x) \wedge \neg B'(x)\) over \(A_s\) returns the desired ICAR-answers, i.e., the set \(\{a, c, f\}\).

Consequently, completion and RL ABox-saturation systems can be used to efficiently compute the ICAR-answers over DL-Lite TBoxes. A minor technical issue is that we first need to remove all inconsistent singleton assertions.

**Definition 8.** Let \(T\) be a L-TBox and \(A\) and \(\text{ans}\) then we denote by \(\text{cr}(A, T)\) the subset of \(A\) that is obtained by removing from \(A\) all assertions \(\alpha \in A\) such that the set \(T \cup \{\alpha\}\) is inconsistent.

Our previous claims are formalised next.

**Theorem 9.** Let \(Q\) be a CQ, let \(T\) be a DL-Lite-TBox, let \(A\) be an ABox and let \(\langle R_Q, R_D \rangle\) be a rewriting for \(Q, T\). Let also \(\text{ans}\) be an RL ABox-saturation system and let \(TC = T^+ \cup C\) a completion of \(T\) for \(\text{ans}\). Then,
\[
\text{cert}^s(Q, T \cup A) = \text{ans}(\text{ref}(R_Q, T), TC \cup \text{cr}(A, T))
\]
Note that for DL-Lite TBoxes, completions for RL systems and rewritings for any conjunctive query always exist. Hence, our result is of high practical relevance.

3.1 Additional Optimisations

Interestingly, as the following example shows, the use of ABox-saturation systems allows us to further simplify the structure of the query returned by the procedure ref.

**Example 10.** Consider the following TBox and ABox:
\[
T = \{ B \subseteq A, B' \subseteq A', A \subseteq \neg A' \}
\]
\[
A = \{ B(a), A'(a), (b), b(b) \}
\]
and consider also the query \(Q = B(x)\). Then, we have \(\text{ref}(Q, T) = \{Q'\}\), where \(Q' = B(x) \wedge \neg A'(x) \wedge \neg B'(x)\) which evaluated over \(A\) returns \(\emptyset\). Note that both of the negative atoms are needed in the query; \(\neg A'(x)\) to exclude individual “\(a\)” and \(\neg B'(x)\) to exclude “\(b\)”.

Consider now an RL ABox-saturation system \(\text{ans}\). For \(\text{ans}\) we have \(C = \emptyset\) and over \(A \cup T^+\) it will compute \(A_s = A \cup \{A(a), A(b), A'(b)\}\). As can be seen, due to \(A'(b) \in A_s\) and \(\neg A'(x)\) in \(Q'\) we can drop atom \(\neg B'(x)\). Indeed, evaluating \(Q'' = B(x) \wedge \neg A'(x)\) over \(A_s\) computes \(\emptyset\) as required. \(\diamondsuit\)

It follows that some negative atoms added by ref can be discarded.

**Definition 11.** Let \(T\) be a L-TBox and let \(Q\) be a CQ. An atom \(\neg A(x)\) in \(Q\) is called covered if there exists some other atom \(\neg B(x)\) in \(Q\) such that \(T \models A \subseteq B\). An atom \(\neg R(x, y)\) in \(Q\) is called covered if there exists some other atom \(\neg A(x), \neg A(y), \neg S(x, y), \text{or } \neg S(y, x)\) in \(Q\) such that \(T \models \exists R \subseteq A, \exists S \subseteq A, R \subseteq S, \text{or } R \subseteq S^{-}\), respectively. We denote with \(\text{min}(Q, T)\) the query derived from \(Q\) by removing all the covered atoms. Moreover, for a UCQ \(R\) we define \(\text{ref}_{\text{min}}(R, T) = \bigcup_{Q' \subseteq \text{ref}(R, T) \setminus \{\text{min}(Q, T)\}}\).

**Theorem 12.** Let \(Q\) be a CQ, let \(T\) be a DL-Lite-TBox, let \(A\) be an ABox and \(\langle R_Q, R_D \rangle\) be a rewriting for \(Q, T\). Let also \(\text{ans}\) be an RL ABox-saturation system and let \(TC = T^+ \cup C\) a completion of \(T\) for \(\text{ans}\). Then,
\[
\text{cert}^s(Q, T \cup A) = \text{ans}(\text{ref}_{\text{min}}(R_Q, T), TC \cup \text{cr}(A, T))
\]
As we will see in the evaluation section, the above minimisation improves performance significantly.
Then, we have saturation system, and let Theorem 15.

\[ \text{Proposition 14.} \]

A completion of assertions that are derived by the saturation procedure.

It performs a consistency check in order to restrict the amount of assertions that are derived by the saturation procedure.

\[ \text{Example 13.} \]

Consider the following E\(\subseteq\)-TBox \(\mathcal{T} = \{A \sqcap B \sqsubseteq C, A \sqcap B \sqsubseteq \bot\} \) and ABox \(\mathcal{A} = \{A(a), B(a)\}\) and let \(Q = C(x)\). Clearly, we have \(\text{clc}(\mathcal{T}, \mathcal{A}) = \mathcal{A}\) and hence \(\text{cert}^{E}(Q, \mathcal{T} \cup \mathcal{A}) = \emptyset\). 

Now, since \(\mathcal{T}\|_{\mathcal{A}} = \{A \sqcap B \subseteq C\}\) any \(\mathcal{RL}\) ABox-saturation system \(\text{ans}\) would compute \(\mathcal{T}^+ \cup \mathcal{A}\) the saturation \(\mathcal{A}_s = \mathcal{A} \cup \{C(a)\}\). Finally, since \(\text{ref}(Q, \mathcal{T}) = \{Q\}\) then we will have \(\text{cert}(\text{ref}(Q, \mathcal{T}), \mathcal{A}) = \{a\}\).

Consequently, in the case of more expressive languages, ABox-saturation systems compute far too many assertions and hence answers. To control them one has to either extend the \(\text{ref}\) procedure to include additional negative atoms (e.g., \(\neg A(x) \land \neg B(x)\) in the previous example) or prevent some assertions from being derived in the first place (e.g., \(C(a)\)).

We follow the second approach and propose Algorithm 1 which saturates the input ABox over the \(\mathcal{RL}\) fragment of the given TBox by also taking into account the negative inclusions. More precisely, the algorithm first translates the \(\mathcal{RL}\) fragment of \(\mathcal{T}^+\) into datalog rules \(\mathcal{P}\) and then saturates \(\mathcal{A}\) using \(\mathcal{P}\). However, every time some rule of \(\mathcal{P}\) “fires” (Line 5), before adding the conclusion \(H\sigma\) to the saturation, in Line 6, it performs a consistency check in order to restrict the amount of assertions that are derived by the saturation procedure.

Clearly, over DL-Lite-TBoxes \(\mathcal{T}\) Algorithm 1 computes \(\text{clc}(\mathcal{T}, \mathcal{A})\) for any \(\mathcal{A}\).

\[ \text{Proposition 14.} \]

Let \(\mathcal{T}\) be a DL-Lite-TBox, let \(\mathcal{A}\) be an ABox, and let \(Q\) be a CQ. Let also \(\text{ans}\) be some \(\mathcal{RL}\) ABox-saturation system and \(\mathcal{C}\) a completion of \(\mathcal{T}\) for \(\text{ans}\). Then, \(\text{clc}(\mathcal{T}, \mathcal{A}) = \text{ApproxAns}(\mathcal{T} \cup \mathcal{C}, \mathcal{A})\).

However, for more expressive DLs the algorithm approximates \(\text{clc}(\mathcal{T}, \mathcal{A})\) from above.

\[ \text{Theorem 15.} \]

Let \(\mathcal{T}\) be a L-TBox, let \(\text{ans}\) be an \(\mathcal{RL}\) ABox-saturation system, and let \(\mathcal{C}\) be a completion of \(\mathcal{T}\) for \(\text{ans}\). Then, we have \(\text{clc}(\mathcal{T}, \mathcal{A}) \subseteq \text{ApproxAns}(\mathcal{T} \cup \mathcal{C}, \mathcal{A})\) and \(\text{ral}\).

Moreover, in contrast to \(\text{clc}\), for many DLs \(\text{clc}^+\) can be computed in polynomial time.

\[ \text{Theorem 19.} \]

Let \(\mathcal{L}\) be some DL such that deciding consistency of ABoxes can be done in polynomial time and \(\mathcal{L}\) is also datalog rewritable. Then, for every L-TBox \(\mathcal{T}\) and ABox \(\mathcal{A}\) \(\text{clc}^+(\mathcal{T}, \mathcal{A})\) can be computed in polynomial time with respect to the size of the data.

The following is an immediate consequence of Theorem 19 and various results in datalog rewritability of tractable Horn-DLs [Calvanese et al., 2007; Pérez-Urbina et al., 2010; Hustadt et al., 2005] as well as highly expressive non-Horn DLs [Kaminski et al., 2014].

\[ \text{Corollary 20.} \]

Let \(\mathcal{L}\) be any of the following DLs:

- DL-Lite, E\(\subseteq\)HI\(\subseteq\)HIQ; or
- \(\mathcal{SHI}^2\) or \(\mathcal{ALCHI}\) that is simple w.r.t. disjunctive predicates [Kaminski et al., 2014].

Then, for every L-TBox \(\mathcal{T}\) and ABox \(\mathcal{A}\), \(\text{clc}^+(\mathcal{T}, \mathcal{A})\) can be computed in polynomial time with respect to the size of \(\mathcal{A}\).
Algorithm 2 ABoxIARRepair(T, A)

Input: TBox T, ABox A. Output: IAR of T ∪ A

1: A := cr(A, T)
2: Φ := ∅
3: for all C ⊆ ¬D ∈ T do add query π(C) ∧ π(D) to Φ
4: for all R ⊆ ¬S ∈ T do add query π(S) ∧ π(R) to Φ
5: for all Qσ ∈ Φ do
6: Compute a UCQ rewriting Rφ for Qσ, T
7: for all Q′σ ∈ Rφ and all σ s.t. Q′σ ⊆ A do
8: Remove Q′σσ from A
9: return A

Concluding this section we note that the output of Algorithm 1 can (possibly) be used to compute some upper approximation of the ICAR-answers. More precisely, for some CQ Q we can compute a rewriting ⟨RQ, R⊙⟩ for Q, T (if it exists) and then calculate cert(ref(RQ, T), A). However, note that ref would also be an approximation as, to the best of our knowledge, it has only been defined for DL-Lite.

5 Efficient QA under the IAR Semantics

Concluding our technical contributions we show that our framework can also be used to compute IAR-answers. More precisely, by the results of Section 3 and the results by Rosati [2011] we have the following.

Proposition 21. Let T be a Ł-TBox, let A be an ABox, let Q be a CQ, and let AIA be the IAR of T ∪ A. Let also ans be an RŁ ABox-saturation system, let C be a completion of T for ans and let ⟨RQ, R⊙⟩ be a rewriting for Q, T. Then, we have certIA(Q, T ∪ A) = ans(RQ, T ∪ C ∪ AIA).

Since completions of DL-Lite and RŁ TBoxes for RŁ ABox-saturation systems always exist [Stoilos et al., 2011] and Rosati [2011] showed that for TBoxes expressed in DL-Lite and RŁ I, the IAR AIA of some ABox can be computed in time polynomial w.r.t. the size of A, it follows that our approach can compute the IAR-answers over these languages effectively by exploiting practically scalable RŁ systems. Unfortunately, even in these cases computing AIA can still be difficult and time-consuming if A is very large.

For DL-Lite TBoxes, an algorithm to compute AIA was proposed and implemented in [Rosati et al., 2012]. This algorithm annotates all assertions in A and then updates their annotation if they violate some negative inclusions of T. All assertions whose annotation changed are finally removed.

Inspired by the techniques in [Lembo et al., 2015] we propose a different algorithm that is based on query rewriting and not on ABox annotation; this is depicted in Algorithm 2. The algorithm builds a special query for each negative inclusion in T; π maps a concept A to A(x), role R to R(x, y) and a concept R to ∃y R(x, y). Then, a rewriting for QA, T is computed and each member of the rewriting is evaluated over A. If some of these queries match to assertions in A, then these should be removed from A as they violate the negative inclusion.

Proposition 22. For a given DL-Lite-TBox T and ABox A, ABoxIARRepair(T, A) returns an IAR of T ∪ A.

Algorithm 2 is amenable to an important optimisation. Instead of evaluating Q′σ over A (Lines 7–8) we can proceed as follows: at a pre-processing step we can evaluate all atomic queries A(x) and R(x, y) separately and cache their answers and the assertions of A that these atoms have a match. Then, to compute the answers of Q′σ over A we can use the pre-computed partial answers of each atom.

6 Evaluation

We implemented our ICAR- (both the standard and the optimised one) and IAR-answering approaches into the prototype system SaQAI2 (Saturation based Query Answering under Inconsistencies); in the following the various versions of SaQAI (standard/optimised ICAR and IAR) are called SaQIC, SaQICO, and SaQIA, respectively. Unfortunately, implementing Algorithm 1 would either require modifying the internals of an ABox-saturation system or implementing one from scratch. Our system uses GraphDB [Kiryakov et al., 2010] as an ABox-saturation system, Hydrowl [Stoilos, 2014] to compute completions, and Rapid [Trivela et al., 2015] for rewriting.

For the evaluation we used the experimental setting proposed in [Bienvenu et al., 2014] which consists of a DL-Lite version of the LUBM20 ontology [Lutz et al., 2013] extended with additional negative inclusions, a set of test queries, and several inconsistent ABoxes. We use the same ABox and query names as in [Bienvenu et al., 2014]. For example, AIC indicates an ABox containing data for n universities and inconsistencies added with probability m. Due to space limitations we will present results only for some of the test queries and for the larger datasets. To the best of our knowledge no available ICAR-answering system exists (we could not obtain the preliminary system reported in [Masotti et al., 2011]), hence we compared against CQApri [Bienvenu et al., 2014] and QuID [Rosati et al., 2012], two IAR-answering systems.

Computing a completion of LUBM20 for GraphDB was done only once and required less than 5 seconds. Moreover, no unsatisfiable concepts were detected, hence cr(AIC, T) also completed in less than a second.

Table 1 presents the results for all considered approaches and tools. In the table we have grouped together number of answers, pre-processing times (note that different systems can perform different pre-processing tasks) and query evaluation. Column Sat denotes loading time of T ∪ C ∪ AIC into GraphDB and computation of the saturation As IC. First, we can note that, as expected, in many queries our ICAR-answering approach returned far more answers than CQApri (SaQIC returned the same IAR-answers as CQApri, however, QuID returned different answers indicating some kind of bug but we did not investigate further). There are actually cases where CQApri returns 0 answers whereas our approach returns some answers to the input query. Second, we can observe that our optimisations in Section 3.1 are indeed quite relevant and improved performance of our ICAR-answering approach significantly. This is because, in most cases refmin (which SaQICO uses) computes queries containing much fewer negative atoms than ref; e.g., in query q1 it computes 32

http://image.ece.ntua.gr/~etsalap/SaQAI
<table>
<thead>
<tr>
<th></th>
<th>Number of Answers</th>
<th>Loading, Pre-processing, and Evaluation (in seconds)</th>
<th>IAR Answering</th>
<th>( A^{\text{op}} )</th>
<th>( q_1 )</th>
<th>255,839</th>
<th>251,991</th>
<th>47.5</th>
<th>4.6</th>
<th>0.2</th>
<th>17.6</th>
<th>11.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A^{\text{op}} )</td>
<td>( q_2 )</td>
<td>88,994</td>
<td>88,816</td>
<td>22.3</td>
<td>1.6</td>
<td>&lt;0.1</td>
<td>12.2</td>
<td>6.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>966,856</td>
<td>769,786</td>
<td>1,060.3</td>
<td>99.3</td>
<td>1.0</td>
<td>14.1</td>
<td>17.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>2,242</td>
<td>2,228</td>
<td>2.4</td>
<td>0.3</td>
<td>0.1</td>
<td>31.2</td>
<td>2.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>189,019</td>
<td>189,519</td>
<td>227.6</td>
<td>21.1</td>
<td>0.9</td>
<td>190.9</td>
<td>61.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>38,244</td>
<td>38,244</td>
<td>33.5</td>
<td>2.2</td>
<td>0.1</td>
<td>84.0</td>
<td>48.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>19.6</td>
<td>52.9</td>
<td>1,437.4</td>
<td>58.1</td>
<td>1.0</td>
<td>1</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>20.1</td>
<td>53.8</td>
<td>2,072.3</td>
<td>65.5</td>
<td>&lt;0.1</td>
<td>9.6</td>
<td>4.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>21.2</td>
<td>55.2</td>
<td>3,205.0</td>
<td>84.6</td>
<td>&lt;0.1</td>
<td>0.7</td>
<td>29.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>59.7</td>
<td>114.2</td>
<td>3,184.3</td>
<td>124.6</td>
<td>&lt;0.1</td>
<td>2.3</td>
<td>4.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>61.2</td>
<td>113.9</td>
<td>5,246.0</td>
<td>136.8</td>
<td>&lt;0.1</td>
<td>0.9</td>
<td>20.2</td>
<td>238.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>66.3</td>
<td>122.1</td>
<td>-</td>
<td>205.7</td>
<td>&lt;0.1</td>
<td>0.2</td>
<td>42.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>66.3</td>
<td>122.1</td>
<td>-</td>
<td>205.7</td>
<td>&lt;0.1</td>
<td>0.2</td>
<td>42.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>66.3</td>
<td>122.1</td>
<td>-</td>
<td>205.7</td>
<td>&lt;0.1</td>
<td>0.2</td>
<td>42.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Results for SaQ\(^{\text{op}}\), SaQ\(^{\text{op}}\), and CQA\( \text{pri} \) (CQA).

whereas ref computes 353 negative atoms. Third, although SaQ\(^{\text{op}}\) computes more answers (the ICAR-answers) than the other systems, it is significantly faster in nearly all queries compared to QuID and CQA\( \text{pri} \) with only noticeable exception query \( q_4 \). Nevertheless, besides the fact that our system computes more involved semantics, for the same query over ABox \( A^{\text{op}} \) CQA\( \text{pri} \) run out of memory and QuID failed to load the dataset within 90 minutes.

Finally, our method for computing the IAR-answers is also significantly more efficient than QuID and CQA\( \text{pri} \). First, our ABox repairing method is several orders of magnitude faster than QuID which is based on the ABox annotation algorithm in [Rosati et al., 2012] (recall that QuID actually did not manage to load the largest ABox \( A_{20}^{\text{op}} \)) and is also much faster than the pre-processing step of CQA\( \text{pri} \). Second, after the repairing step and due to Proposition 21 our system computes the IAR-answers by standard query evaluation using GraphDB (since the ABox is repaired the ref procedure is not required) and this takes in all cases just a few milliseconds in contrast to QuID and CQA\( \text{pri} \) that still require several seconds to evaluate the queries they have computed and construct the IAR-answers. All in all both our approaches are much more efficient and robust.

7 Conclusions

We proposed a framework for efficient and scalable IAR- and ICAR-answering over inconsistent Description Logic knowledge bases which is based on mature data saturation technologies. The approach is exact for both semantics over DL-Lite and for the IAR semantics over \( \mathcal{E}^\perp \) ontologies. For more expressive DLs we proposed an algorithm that computes an approximation of ICAR-answers but which characterises a new family of semantics that can be computed in polynomial time for a very large family of DLs. Our experiments provided with encouraging results as both of our approaches are more efficient and robust compared to the state-of-the-art.
References


[Kiryakov et al., 2010] Atanas Kiryakov, Barry Bishop, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. The features of BigOWLIM that enabled the BBC’s World Cup website. In Workshop on Semantic Data Management (SemData), 2010.


