Practical ABox cleaning in DL-Lite (progress report)

Giulia Masotti, Riccardo Rosati, Marco Ruzzi

Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Sapienza Università di Roma
Via Ariosto 25, I-00185 Roma, Italy

1 Introduction

One of the most important current issues in Description Logic (DL) ontology management is dealing with inconsistency, that is, the presence of contradictory information in the ontology [7]. It is well-known that the classical semantics of DLs is not *inconsistency-tolerant*, i.e., it does not allow for using in a meaningful way any piece of information in an inconsistent ontology. On the other hand, the size of ontologies used by real applications is scaling up, and ontologies are increasingly merged and integrated into larger ontologies: the probability of creating inconsistent ontologies is consequently getting higher and higher.

In this paper we focus on ABox inconsistency, i.e., the case of inconsistent KBs where the TBox is consistent while the ABox is inconsistent with the TBox, i.e., a subset of the assertions in the ABox contradicts a TBox assertion (or a subset of the TBox). In particular, we are interested in defining a form of automatic *ABox cleaning*, i.e., given $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we want to identify an ABox $\mathcal{A}'$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is consistent and $\mathcal{A}'$ is "as close as possible" to $\mathcal{A}$.

The kind of ABox cleaning we adopt is formally based on inconsistency-tolerant semantics, which overcome the limitations of the classical DL semantics in inconsistency management. In particular, we consider inconsistency-tolerant semantics for general DLs recently proposed in [4], called *IAR semantics* and $ICAR$ *semantics*, for which reasoning has been studied in the context of the Description Logics of the *DL-Lite* family. The notion of ABox repair in the IAR semantics is very simple: the ABox repair of a DL ontology is the intersection of all the maximal subsets of the ABox that are consistent with the TBox. The notion of ABox repair in the $ICAR$ semantics is a variant of the IAR semantics that is based on a notion of "equivalence under consistency" of ABoxes inconsistent with respect to a given TBox. In [4] it was proved that computing the ABox repair of a *DL-Lite*$_A$ ontology is tractable both under IAR semantics and $ICAR$ semantics.

We argue that the results of [4] are very important from the practical viewpoint, for the following reasons: (i) they provide (to the best of our knowledge) the first formally grounded notion of ABox cleaning. In other words, IAR and $ICAR$ are the first inconsistency-tolerant semantics that allow for expressing ABox repairs in terms of a single ABox; (ii) they identify (to the best of our knowledge) the first tractable inconsistency-tolerant semantics in DLs. This paper starts from the above results, and tries to provide an experimental validation that ABox cleaning based on the above semantics is actually feasible. More precisely, we provide the following contributions:

(1) We present effective techniques for ABox cleaning in *DL-Lite*$_A$ under IAR and $ICAR$ semantics. To this aim, we present the Quonto ABox Cleaner (QuAC), which implements, within the Quonto system,[1] techniques for the computation of the ABox repair of a *DL-Lite*$_A$ knowledge base under the above semantics. QuAC constitutes (to the best of our knowledge) the first implementation of a tractable ABox cleaning algorithm for DL ontologies. Moreover, since Quonto delegates the management of the ABox to a relational database system (DBMS), all modifications of the ABox are delegated to the DBMS through SQL queries and updates. This potentially allows for handling and cleaning very large ABoxes.
(2) We report on the experimental analysis that we are actually conducting using QuAC. Our first results are allowing us to understand the actual impact, w.r.t. the efficiency of ABox cleaning, of the different aspects involved in the computation of the ABox repair, and the limits and possibilities of the approach implemented in QuAC.

The paper that is closer to our work is [3], which also presents a technique for ABox cleaning in DL ontologies. However, there are two main differences with our approach: (i) [3] considers the very expressive DL $\mathcal{SHIN}$, in which all the semantics considered by our approach are intractable ([6]); (ii) the two approaches are based on different semantics: in particular, the ABox cleaning algorithm of [3] computes a consistent subset of the ABox which in general is uncomparable with the ABox repair defined by the IAR semantics (and the $ICAR$ semantics).

The rest of the paper is organized as follows. In Section 2, we give some preliminaries, and in particular we introduce *DL-Lite*$_A$ and the definition of the IAR and the $ICAR$ semantics. In Section 3, we present detailed algorithms for ABox cleaning in *DL-Lite*$_A$. In Section 4 we present the QuAC system and report on the experiments we are currently conducting with QuAC. Finally, in Section 5 we conclude the paper.

2 Preliminaries

2.1 The DL *DL-Lite*$_A$

In this paper we consider DL ontologies (knowledge bases) specified in *DL-Lite*$_A$, a member of the *DL-Lite* family of tractable Description Logics [2, 1], which is at the basis of OWL 2 QL, one of the profile of OWL 2, the official knowledge base specification language of the World-Wide-Web Consortium (W3C). *DL-Lite*$_A$ distinguishes concepts from *value-domains*, which denote sets of (data) values, and roles from *attributes*, which denote binary relations between objects and values. Concepts, roles, attributes, and value-domains in this DL are formed according to the following syntax:

$$\begin{array}{ll} B \longrightarrow A \mid \exists Q \mid \delta(U) & E \longrightarrow \rho(U) \\ C \longrightarrow B \mid \neg B & F \longrightarrow \top_D \mid T_1 \mid \cdots \mid T_n \\ Q \longrightarrow P \mid P^- & V \longrightarrow U \mid \neg U \\ R \longrightarrow Q \mid \neg Q & \end{array}$$

In such rules, A, P, and U respectively denote an atomic concept (i.e., a concept name), an atomic role (i.e., a role name), and an attribute name, P^- denotes the inverse of an atomic role, whereas B and Q are called basic concept and basic role, respectively.

[1] `http://www.dis.uniroma.it/~quonto`

Furthermore, $\delta(U)$ denotes the *domain* of U, i.e., the set of objects that U relates to values; $\rho(U)$ denotes the *range* of U, i.e., the set of values that U relates to objects; $\top_D$ is the universal value-domain; $T_1, \ldots, T_n$ are n pairwise disjoint unbounded value-domains.

A *DL-Lite*$_A$ knowledge base (KB) is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is the TBox and $\mathcal{A}$ the ABox. The TBox $\mathcal{T}$ is a finite set of assertions of the form

$$B \sqsubseteq C \qquad Q \sqsubseteq R \qquad E \sqsubseteq F \qquad U \sqsubseteq V \qquad (\mathsf{funct}\ Q) \qquad (\mathsf{funct}\ U)$$

From left to right, the first four assertions respectively denote inclusions between concepts, roles, value-domains, and attributes. In turn, the last two assertions denote functionality on roles and on attributes. In fact, in *DL-Lite*$_A$ TBoxes we further impose that roles and attributes occurring in functionality assertions cannot be specialized (i.e., they cannot occur in the right-hand side of inclusions). Let B_1 and B_2 be basic concepts, and let Q_1 and Q_2 be basic roles. We call *positive inclusions (PIs)* assertions of the form $B_1 \sqsubseteq B_2$, and of the form $Q_1 \sqsubseteq Q_2$, whereas we call *negative inclusions (NIs)* assertions of the form $B_1 \sqsubseteq \neg B_2$ and $Q_1 \sqsubseteq \neg Q_2$.

A *DL-Lite*$_A$ ABox $\mathcal{A}$ is a finite set of membership assertions (ABox assertions) of the forms $A(a)$, $P(a, b)$, and $U(a, v)$, where A, P, and U are as above, a and b belong to Γ_O, the subset of Γ_C containing object constants, and v belongs to Γ_V, the subset of Γ_C containing value constants, where $\{\Gamma_O, \Gamma_V\}$ is a partition of Γ_C.

The semantics of a *DL-Lite*$_A$ knowledge base is given in terms of first-order logic (FOL) interpretations in the usual way. An interpretation $\mathcal{I}$ satisfying a knowledge base $\mathcal{K}$ a called a *model* for $\mathcal{K}$. In the following *Mod*$(\langle \mathcal{T}, \mathcal{A} \rangle)$ will indicate the set of models of the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. A knowledge base $\mathcal{K}$ is satisfiable if it has at least a model, otherwise it is called unsatisfiable. Given an assertion α (which is either a TBox or ABox assertion), we write $\mathcal{K} \models \alpha$ if α is satisfied in every model for $\mathcal{K}$.

Given a TBox $\mathcal{T}$ and an ABox $\mathcal{A}'$, $\mathcal{A}'$ is called a *minimal conflict set for* $\mathcal{T}$ if the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is unsatisfiable and, for every ABox $\mathcal{A}''$ such that $\mathcal{A}'' \subset \mathcal{A}'$, the KB $\langle \mathcal{T}, \mathcal{A}'' \rangle$ is satisfiable. A minimal conflict set for $\mathcal{T}$ is called *unary* if its cardinality (that is, the number of assertions it contains) is 1 and is called *binary* if its cardinality is 2.

2.2 Inconsistency-tolerant semantics for DLs

In this section we recall the inconsistency-tolerant semantics for general DL knowledge bases defined in [4].[2] We assume that, for a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\mathcal{T}$ is satisfiable, whereas $\mathcal{A}$ may be inconsistent with $\mathcal{T}$, i.e., the set of models of $\mathcal{K}$ may be empty.

AR-semantics The first notion of repair that we consider, called *AR-repair*, is a very natural one: a repair is a maximal subset of the ABox that is consistent with the TBox. Thus, an AR-repair is obtained by throwing away from $\mathcal{A}$ a minimal set of assertions to make it consistent with $\mathcal{T}$.

Definition 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. An AR-*repair *of $\mathcal{K}$ is a set $\mathcal{A}'$ of membership assertions such that: (i)$\mathcal{A}' \subseteq \mathcal{A}$; (ii) Mod$(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$; (iii) there does not exist*

[2] Due to space limitations, we refer the reader to [4] for introductory examples illustrating these semantics.

*$\mathcal{A}''$ such that $\mathcal{A}' \subset \mathcal{A}'' \subseteq \mathcal{A}$ and Mod$(\langle \mathcal{T}, \mathcal{A}'' \rangle) \neq \emptyset$. The set of AR-repairs for $\mathcal{K}$ is denoted by AR-Rep$(\mathcal{K})$. Moreover, we say that a first-order sentence ϕ is AR-*entailed *by $\mathcal{K}$, written $\mathcal{K} \models_{AR} \phi$, if $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$ for every $\mathcal{A}' \in$ AR-Rep$(\mathcal{K})$.*

CAR-semantics We start by formally introducing a notion of "equivalence under consistency" for inconsistent KBs.

Given a KB $\mathcal{K}$, let $\mathcal{S}_K$ denote the signature of $\mathcal{K}$, i.e., the set of concept, role, and individual names occurring in $\mathcal{K}$. Given a signature $\mathcal{S}$, we denote with *HB*$(\mathcal{S})$ the *Herbrand Base of* $\mathcal{S}$, i.e. the set of ABox assertions (ground atoms) that can be built over the signature $\mathcal{S}$. Then, given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we define the *consistent logical consequences of* $\mathcal{K}$ as the set *clc*$(\mathcal{K}) = \{\alpha \mid \alpha \in$ *HB*$(\mathcal{S}_K)$ and there exists $\mathcal{A}' \subseteq \mathcal{A}$ such that *Mod*$(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$ and $\langle \mathcal{T}, \mathcal{A}' \rangle \models \alpha\}$. Finally, we say that two KBs $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}, \mathcal{A}' \rangle$ are *consistently equivalent (C-equivalent)* if *clc*$(\langle \mathcal{T}, \mathcal{A} \rangle) =$ *clc*$(\langle \mathcal{T}, \mathcal{A}' \rangle)$.

We argue that the notion of C-equivalence is very reasonable in settings in which the ABox (or at least a part of it) has been "closed" (in a complete or partial way) with respect to the TBox, e.g., when (some or all) the ABox assertions that are entailed by the ABox and the TBox have been added to the original ABox. This may happen, for example, when the ABox is obtained by integrating different (and locally consistent) sources, since some of these sources might have been locally closed with respect to some TBox axioms: this is very likely, for instance, if a source is an RDF graph with RDFS predicates, since many RDF systems materialize in the RDF graph the implicit triples due to the RDFS predicates.

In settings where C-equivalence makes sense, the AR-semantics is not suited to handle inconsistency. In fact, we would expect two C-equivalent KBs to produce the same logical consequences under inconsistency-tolerant semantics. Unfortunately, the AR-semantics does not have this property. A simple example is the following: let $\mathcal{T} = \{$*student* $\sqsubseteq$ $young$, *student* $\sqsubseteq \neg worker\}$ and let $\mathcal{A} = \{$*student*$(mary), worker(mary)\}$, $\mathcal{A}' = \{$*student*$(mary), worker(mary), young(mary)\}$. It is immediate to verify that if $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, then *clc*$(\mathcal{K}) =$ *clc*$(\mathcal{K}') = \mathcal{A}'$, thus $\mathcal{K}$ and $\mathcal{K}'$ are C-equivalent, however $\mathcal{K}' \models_{AR} young(mary)$ while $\mathcal{K} \not\models_{AR} young(mary)$.

To overcome the above problem, the CAR-semantics has been defined in [4], through a modification of the AR-semantics.[3]

Definition 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. A CAR-*repair *for $\mathcal{K}$ is a set $\mathcal{A}'$ of membership assertions such that $\mathcal{A}'$ is an AR-repair of $\langle \mathcal{T}, clc(\mathcal{K}) \rangle$. The set of CAR-repairs for $\mathcal{K}$ is denoted by CAR-Rep$(\mathcal{T}, \mathcal{A})$. Moreover, we say that a first-order sentence ϕ is CAR-*entailed *by $\mathcal{K}$, written $\mathcal{K} \models_{CAR} \phi$, if $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$ for every $\mathcal{A}' \in$ CAR-Rep$(\mathcal{K})$.*

Going back to the previous example, it is immediate to see that, since $\mathcal{K}$ and $\mathcal{K}'$ are C-equivalent, the set of CAR-repairs (and hence the set of CAR-models) of $\mathcal{K}$ and $\mathcal{K}'$ coincide. As the above example shows, there are sentences entailed by a KB under CAR-semantics that are not entailed under AR-semantics. Conversely, it is shown in

[3] The definition provided here of the CAR-semantics is a slight simplification of the one appearing in [4]: this modification, however, does not affect any of the computational results presented in [4].

[4] that the AR-semantics is a sound approximation of the CAR-semantics, i.e., for every KB $\mathcal{K}$ and every FOL sentence ϕ, $\mathcal{K} \models_{AR} \phi$ implies $\mathcal{K} \models_{CAR} \phi$.

IAR-semantics and ICAR-semantics We then recall the IAR-semantics and $ICAR$-semantics, which are sound approximations of the AR-semantics and the CAR-semantics, respectively [4].

Definition 3. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. Then: (i) The IAR-*repair *for $\mathcal{K}$, denoted by IAR-Rep$(\mathcal{K})$ is defined as IAR-Rep$(\mathcal{K}) = \bigcap_{\mathcal{A}' \in \text{AR-Rep}(\mathcal{K})} \mathcal{A}'$. (ii) The $ICAR$-*repair *for $\mathcal{K}$, denoted by ICAR-Rep$(\mathcal{K})$ is defined as ICAR-Rep$(\mathcal{K}) = \bigcap_{\mathcal{A}' \in \text{CAR-Rep}(\mathcal{K})} \mathcal{A}'$. (iii) We say that a first-order sentence ϕ is IAR-*entailed *(respectively, $ICAR$-*entailed*) by $\mathcal{K}$, and we write $\mathcal{K} \models_{IAR} \phi$ (respectively, $\mathcal{K} \models_{ICAR} \phi$), if $\langle \mathcal{T}, \text{IAR-Rep}(\mathcal{K}) \rangle \models \phi$ (respectively, $\langle \mathcal{T}, \text{ICAR-Rep}(\mathcal{K}) \rangle \models \phi$).*

Example 1. Let us consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ where the TBox $\mathcal{T}$ is the following:

$$\mathcal{T} = \{A \sqsubseteq C, B \sqsubseteq C, \exists R \sqsubseteq B, \exists R^- \sqsubseteq D,\ A \sqsubseteq \neg B, (\mathsf{funct}\ R)\}$$

and the ABox $\mathcal{A}$ is $\mathcal{A} = \{A(a), B(a), C(a), R(a,b)\}$. Such a KB is unsatisfiable, due to the presence of the assertions $A(a)$ and $B(a)$ which violate the disjointness assertion in $\mathcal{T}$. The following are the standard AR-repairs of $\mathcal{A}$:

$$\mathcal{A}_{AR}{}^1 = \{B(a), C(a), R(a,b)\}, \quad \mathcal{A}_{AR}{}^2 = \{A(a), C(a)\}$$

Then, we have $clc(A) = \{A(a), B(a), C(a), R(a,b), D(a)\}$. Therefore, the CAR-repair of $\mathcal{A}$ are as follows:

$$\mathcal{A}_{CAR}{}^1 = \{B(a), C(a), R(a,b), D(b)\}, \quad \mathcal{A}_{CAR}{}^2 = \{A(a), C(a), D(b)\}$$

Consequently, the IAR-repair and ICAR-repair are the following:

$$\mathcal{A}_{IAR} = \mathcal{A}_{AR}{}^1 \cap \mathcal{A}_{AR}{}^2 = \{C(a)\},\ \ \mathcal{A}_{ICAR} = \mathcal{A}_{CAR}{}^1 \cap \mathcal{A}_{CAR}{}^2 = \{C(a), D(a)\}$$

Example 2. One might conjecture that the IAR semantics collapses into a simple ABox cleaning technique which deletes from the ABox all the assertions that participate in conflicts with the TBox. This is actually not the case, because, as explained in [4], the IAR-repair actually deletes only the assertions that participate in *minimal* conflict sets. Here is an example: given the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with $\mathcal{T} = \{A \sqsubseteq \neg \exists R,\ R \sqsubseteq \neg R^-\}$, $\mathcal{A} = \{A(a),\ R(a,a)\}$, the IAR-repair of $\mathcal{K}$ is $\{A(a)\}$. That is, the assertion $A(a)$ belongs to the IAR-repair even if it participates in the conflict set $\{A(a),\ R(a,a)\}$ caused by the concept disjointness $A \sqsubseteq \neg \exists R$: the reason is that such a conflict set is not minimal because of the unary conflict set $\{R(a,a)\}$ caused by the role disjointness $R \sqsubseteq \neg R^-$.

3 Algorithms for ABox cleaning

The technique for computing the ICAR-repair of a *DL-Lite*$_A$ ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ is based on the idea of deleting from $\mathcal{A}$ all the membership assertions participating in *minimal* conflict sets for $\mathcal{T}$. As shown in [4], this task is relatively easy (in particular, tractable) in *DL-Lite*$_A$ because the following property holds: for every *DL-Lite*$_A$ TBox $\mathcal{T}$, all the minimal conflict sets for $\mathcal{T}$ are either unary conflict sets or binary conflict sets.

This property is actually crucial for tractability of reasoning under IAR and $ICAR$ semantics. As shown in [6] this property is not shared by other tractable DLs (e.g. $\mathcal{EL}_\bot$), in which the size of minimal conflict sets is not bound by a constant but depends on the size of the ABox.

We now present detailed algorithms for computing the IAR-repair and the $ICAR$-repair of a *DL-Lite*$_A$ ontology. These algorithms exploits the techniques presented in [4], whose aim was only to provide PTIME upper bounds for the problem of computing such repairs. In particular, the present algorithms specify efficient ways of detecting minimal conflict sets and computing consistent logical consequences. Instead, the previous techniques check all unary and binary subsets of the ABox for these purposes.

In the following, we call *annotated ABox assertion* an expression ξ of the form $\langle \alpha, \gamma \rangle$ where α is an ABox assertion and γ is a value in the set $\{\textit{cons}, \textit{ucs}, \textit{bcs}\}$. Furthermore, we call *annotated ABox* a set of annotated ABox assertions. The intuition behind an annotated ABox assertion ξ is that its annotation γ expresses whether the associated ABox expression α does not participate in any minimal conflict set (*cons*) or participates in a unary conflict set (*ucs*) or to a binary conflict set (*bcs*).

The following algorithm *QuAC-ICAR* computes the $ICAR$-repair of a *DL-Lite*$_A$ KB. For ease of exposition, the algorithm does not report details on the treatment of attributes, which are actually handled in a way analogous to roles.

```
**Algorithm** QuAC-ICAR($\mathcal{K}$)
**input**: DL-Lite$_A$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$,   **output**: $ICAR$-repair of $\mathcal{K}$
**begin**
// STEP 1: create annotated ABox $\mathcal{A}_{ann}$
  $\mathcal{A}_{ann} = \emptyset$;
  **for each** $\alpha \in \mathcal{A}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \langle \alpha, \textit{cons} \rangle$;
// STEP 2: detect unary conflict sets in $\mathcal{A}_{ann}$
  **for each** concept name $A$ s.t. $\mathcal{T} \models A \sqsubseteq \neg A$ **do**
      **for each** $\xi = \langle A(a), \textit{cons} \rangle \in \mathcal{A}_{ann}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi\} \cup \{\langle A(a), \textit{ucs} \rangle\}$;
  **for each** role name $R$ s.t. $\mathcal{T} \models R \sqsubseteq \neg R$ **do**
      **for each** $\xi = \langle R(a,b), \textit{cons} \rangle \in \mathcal{A}_{ann}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi\} \cup \{\langle R(a,b), \textit{ucs} \rangle\}$;
  **for each** role name $R$ s.t. $\mathcal{T} \models R \sqsubseteq \neg R^-$ or $\mathcal{T} \models \exists R \sqsubseteq \neg \exists R^-$ **do**
      **for each** $\xi = \langle R(a,a), \textit{cons} \rangle \in \mathcal{A}_{ann}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi\} \cup \{\langle R(a,a), \textit{ucs} \rangle\}$;
// STEP 3: compute consistent logical consequences in $\mathcal{A}_{ann}$
  **for each** inclusion $A \sqsubseteq B$ with $A, B$ atomic concepts such that $\mathcal{T} \models A \sqsubseteq B$ **do**
      **for each** $\langle A(a), \gamma \rangle \in \mathcal{A}_{ann}$ such that $\gamma \neq \textit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle B(a), \textit{cons} \rangle\}$;
  **for each** inclusion $\exists R \sqsubseteq A$ with $A$ atomic concept such that $\mathcal{T} \models \exists R \sqsubseteq A$ **do**
      **for each** $\langle R(a,b), \gamma \rangle \in \mathcal{A}_{ann}$ such that $\gamma \neq \textit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle A(a), \textit{cons} \rangle\}$;
  **for each** inclusion $\exists R^- \sqsubseteq A$ with $A$ atomic concept such that $\mathcal{T} \models \exists R^- \sqsubseteq A$ **do**
      **for each** $\langle R(a,b), \gamma \rangle \in \mathcal{A}_{ann}$ and $\gamma \neq \textit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle A(b), \textit{cons} \rangle\}$;
  **for each** inclusion $R \sqsubseteq S$ with $R, S$ atomic roles such that $\mathcal{T} \models R \sqsubseteq S$ **do**
      **for each** $\langle R(a,b), \gamma \rangle \in \mathcal{A}_{ann}$ and $\gamma \neq \textit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle S(a,b), \textit{cons} \rangle\}$;
  **for each** inclusion $R^- \sqsubseteq S$ with $R, S$ atomic roles such that $\mathcal{T} \models R^- \sqsubseteq S$ **do**
      **for each** $\langle R(b,a), \gamma \rangle \in \mathcal{A}_{ann}$ and $\gamma \neq \textit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle S(a,b), \textit{cons} \rangle\}$;
// STEP 4: detect binary conflict sets in $\mathcal{A}_{ann}$
  **for each** disjointness $A \sqsubseteq \neg B$ with $A, B$ atomic concepts
      such that $\mathcal{T} \models A \sqsubseteq \neg B$ **do**
      **for each** pair $\xi_1 = \langle A(a), \gamma_1 \rangle, \xi_2 = \langle B(a), \gamma_2 \rangle \in \mathcal{A}'_{ann}$
```

```
            such that $\gamma_1, \gamma_2 \neq ucs$ do
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle A(a), bcs\rangle, \langle B(a), bcs\rangle\}$;
    for each disjointness $A \sqsubseteq \neg\exists R$ with $A$ atomic concept
        such that $\mathcal{T} \models A \sqsubseteq \neg\exists R$ do
        for each pair $\xi_1 = \langle A(a), \gamma_1\rangle, \xi_2 = \langle R(a,b), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $\gamma_1, \gamma_2 \neq ucs$ do
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle A(a), bcs\rangle, \langle R(a,b), bcs\rangle\}$;
    for each disjointness $A \sqsubseteq \neg\exists R^-$ with $A$ atomic concept
        such that $\mathcal{T} \models A \sqsubseteq \neg\exists R$ do
        for each pair $\xi_1 = \langle A(a), \gamma_1\rangle, \xi_2 = \langle R(b,a), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $\gamma_1, \gamma_2 \neq ucs$ do
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle A(a), bcs\rangle, \langle R(b,a), bcs\rangle\}$;
    for each disjointness $R \sqsubseteq \neg S$ with $R, S$ atomic roles
        such that $\mathcal{T} \models R \sqsubseteq \neg S$ do
        for each pair $\xi_1 = \langle R(a,b), \gamma_1\rangle, \xi_2 = \langle S(a,b), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $\gamma_1, \gamma_2 \neq ucs$ do
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle R(a,b), bcs\rangle, \langle S(a,b), bcs\rangle\}$;
    for each disjointness $R \sqsubseteq \neg S^-$ with $R, S$ atomic roles
        such that $\mathcal{T} \models R \sqsubseteq \neg S^-$ do
        for each pair $\xi_1 = \langle R(a,b), \gamma_1\rangle, \xi_2 = \langle S(b,a), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $\gamma_1, \gamma_2 \neq ucs$
            do $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle R(a,b), bcs\rangle, \langle S(b,a), bcs\rangle\}$;
    for each functionality assertion $(\mathsf{funct}\ R) \in \mathcal{T}$ with $R$ atomic role do
        for each pair $\xi_1 = \langle R(a,b), \gamma_1\rangle, \xi_2 = \langle R(a,c), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $b \neq c$ and $\gamma_1, \gamma_2 \neq ucs$ do
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle R(a,b), bcs\rangle, \langle R(a,c), bcs\rangle\}$;
    for each functionality assertion $(\mathsf{funct}\ R^-) \in \mathcal{T}$ with $R$ atomic role do
        for each pair $\xi_1 = \langle R(b,a), \gamma_1\rangle, \xi_2 = \langle R(c,a), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $b \neq c$ and $\gamma_1, \gamma_2 \neq ucs$ do
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle R(b,a), bcs\rangle, \langle R(c,a), bcs\rangle\}$;
// STEP 5: extract the ICAR repair from $\mathcal{A}_{ann}$
    $\mathcal{A}' = \emptyset$;
    for each $\langle \alpha, cons\rangle \in \mathcal{A}_{ann}$ do $\mathcal{A}' = \mathcal{A}' \cup \{\alpha\}$;
    return $\mathcal{A}'$
end
```

The algorithm *QuAC-ICAR* consists of five steps which can be informally described as follows.

step 1 *copy of $\mathcal{A}$ into an annotated ABox $\mathcal{A}_{ann}$*. In this step, the value of the annotation is initialized to *cons* for all ABox assertions.

step 2 *detection of the unary conflict sets in $\mathcal{A}_{ann}$*. For every assertion of the form $\xi = \langle \alpha, cons\rangle$, such that $\{\alpha\}$ is a unary conflict set for $\mathcal{T}$, $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi\} \cup \{\langle \alpha, ucs\rangle\}$, i.e., the annotation relative to α is changed to *ucs*. Unary conflict sets are actually detected through TBox reasoning, by looking at empty concepts and roles in $\mathcal{T}$, as well as asymmetric roles, i.e., roles disjoint with their inverse.

step 3 *computation of the consistent logical consequences in $\mathcal{A}_{ann}$*. Here, the task is to compute all ABox assertions that are entailed by $\mathcal{T}$ together with any $\mathcal{T}$-consistent subset of $\mathcal{A}$. In *DL-Lite$_A$*, this actually corresponds to computing the ABox assertions that are entailed by $\mathcal{T}$ together with the ABox obtained from $\mathcal{A}$ by deleting all unary conflict sets for $\mathcal{T}$. Hence, what the algorithms does is computing the ABox assertions that are logical consequence of $\mathcal{T}$ and of the assertions of $\mathcal{A}_{ann}$ which have not been annotated as unary conflict sets in the previous step.

step 4 *detection of the binary conflict sets in $\mathcal{A}_{ann}$.* For every pair of assertions of the form $\xi_1 = \langle \alpha_1, \gamma_1 \rangle$, $\xi_2 = \langle \alpha_2, \gamma_2 \rangle$ such that $\gamma_1 \neq$ *ucs* and $\gamma_2 \neq$ *ucs* and $\{\alpha, \beta\}$ is a binary conflict set for $\mathcal{T}$, $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle \alpha, \textit{bcs} \rangle, \langle \beta, \textit{bcs} \rangle\}$, i.e., the annotation relative to α and β is changed to *bcs*. As in the case of unary conflict sets, to find binary conflict sets the algorithm looks for disjoint concepts and roles in $\mathcal{T}$, as well as functional roles.

step 5 *extraction of the ICAR-repair from $\mathcal{A}_{ann}$.* The $ICAR$-repair can be now simply extracted from the annotated ABox $\mathcal{A}_{ann}$, by eliminating both unary conflict sets and binary conflict sets. Therefore, for every assertion of the form $\langle \alpha, \textit{cons} \rangle$ in $\mathcal{A}_{ann}$, α is copied into the (non-annotated) ABox $\mathcal{A}'$ which is finally returned by the algorithm.

The algorithm *QuAC-IAR* is very similar to *QuAC-ICAR*: the only difference is that it does not execute step 3, i.e., computation of consistent logical consequences. Correctness of the above algorithms can be proved starting from the results in [4].

Theorem 1. *Let $\mathcal{K}$ be a DL-Lite$_A$ KB and let $\mathcal{A}'$ be the ABox returned by QuAC-ICAR$(\mathcal{K})$. Then, $\mathcal{A}' =$ ICAR-Rep$(\mathcal{K})$. Moreover, let $\mathcal{A}''$ be the ABox returned by QuAC-IAR$(\mathcal{K})$. Then, $\mathcal{A}'' =$ IAR-Rep$(\mathcal{K})$.*

4 Implementation and experiments

We have implemented the above algorithms *QuAC-ICAR* and *QuAC-IAR* within the Quonto system, in a module called QuAC (the Quonto ABox Cleaner). Essentially, QuAC is a Java implementation of the above algorithms where operations on the involved ABoxes are delegated to a relational database system (DBMS). In fact, in the Quonto architecture, the management of the ABox is delegated to a DBMS: therefore, all the operations on ABox assertions of the algorithms for computing repairs are executed in QuAC by the DBMS used by Quonto, through appropriate SQL scripts.

We are currently experimenting QuAC in order to answer several open questions, among which:

– the computational cost of the various steps of the algorithm *QuAC-IAR* and *QuAC-ICAR*;
– the scalability of such algorithms;
– measuring the difference in terms of computational costs of the IAR semantics and the $ICAR$ semantics;
– the impact of the "degree of inconsistency" of the ABox on the computational cost of the algorithms.

The tables reported in Figure 1 and Figure 2 present some of the experimental results that we have obtained so far. The TBox used in the experiments has 30 concept names, 20 role names, 10 attribute names, and about 200 TBox assertions. The various ABoxes used have been automatically generated.

The first table presents the experimental results for a version of Quonto that uses a main memory database (H2) to handle the ABox, while the second table presents the same results when Quonto uses a standard (disk-resident) database (PostgreSQL). The results have been conducted on a Pentium i7 (2.67 GHz) CPU with 6GB RAM under Windows 7 (64 bit) operating system. We have also executed the same tests using the MySQL DBMS, with results analogous to those obtained with PostgreSQL.

In the tables, the first column reports the number of assertions in the ABox, while the second column reports the percentage of ABox assertions that participate in minimal conflict sets for the considered TBox. Moreover:

- Δ_1 denotes the time to create the annotated ABox;
- Δ_2^{IAR} denotes the time to detect unary and binary conflict sets;
- Δ_3^{IAR} denotes the time to extract the IAR-repair from the annotated ABox;
- Δ_2^{ICAR} denotes the time to detect unary conflict sets, compute consistent logical consequences and detect binary conflict sets;
- Δ_3^{ICAR} denotes the time to extract the $ICAR$-repair from the annotated ABox;
- Δ^{IAR} is the total time to compute the IAR-repair, i.e., $\Delta_1 + \Delta_2^{IAR} + \Delta_3^{IAR}$;
- Δ^{ICAR} is the total time to compute the $ICAR$-repair, i.e., $\Delta_1 + \Delta_2^{ICAR} + \Delta_3^{ICAR}$;
- all times are expressed in milliseconds.

ABox size	% incons.	Δ_1	Δ_2^{IAR}	Δ_3^{IAR}	Δ^{IAR}	Δ_2^{ICAR}	Δ_3^{ICAR}	Δ^{ICAR}
$1,000$	1%	188	296	109	593	749	250	1,187
$1,000$	5%	188	358	78	624	749	250	1,187
$1,000$	10%	188	296	94	578	749	266	1,203
$3,000$	1%	359	670	251	1,280	1,997	266	2,622
$3,000$	5%	359	795	234	1,388	1,997	251	2,607
$3,000$	10%	359	717	126	1,202	1,997	282	2,638
$10,000$	1%	515	874	141	1,530	3,495	1,424	5,434
$10,000$	5%	515	781	171	1,467	3,495	1,376	5,386
$10,000$	10%	515	982	172	1,669	3,495	1,156	5,166
$30,000$	1%	812	3,075	422	4,309	22,635	3,559	27,006
$30,000$	5%	812	3,355	418	4,585	22,635	2,498	25,945
$30,000$	10%	812	3,417	344	4,573	22,635	2,748	26,195

Fig. 1. Results for main memory database (H2)

The above experimental results show that:

(i) with both the main memory DB and the disk-resident DB, the computation of the IAR-repair (Δ^{IAR} column) seems really scalable, while the computation of the $ICAR$-repair suffers from the additional step of computing logical consequences, which is computationally very expensive: its cost actually dominates the cost of all the other steps;

(ii) the percentage of inconsistency, i.e., the fraction of ABox assertions that participate in minimal conflict sets, does not have a significant impact on the execution time of both algorithms;

ABox size	% incons.	Δ_1	Δ_2^{IAR}	Δ_3^{IAR}	Δ^{IAR}	Δ_2^{ICAR}	Δ_3^{ICAR}	Δ^{ICAR}
$1,000$	1%	718	516	5,117	6,351	1,358	6,314	7,672
$1,000$	5%	718	515	5,258	6,491	1,358	6,070	7,428
$1,000$	10%	718	531	4,680	5,929	1,358	5,929	7,287
$3,000$	1%	1,840	688	5,444	7,972	4,011	8,747	12,758
$3,000$	5%	1,840	750	5,366	7,956	4,011	7,317	11,328
$3,000$	10%	1,840	797	5,304	7,941	4,011	8,284	12,295
$10,000$	1%	5,850	1,171	10,235	17,256	16,990	19,115	36,105
$10,000$	5%	5,850	1,499	10,297	17,646	16,990	19,424	36,414
$10,000$	10%	5,850	1,561	9,923	17,334	16,990	17,926	34,916
$30,000$	1%	16,255	3,823	20,702	40,780	134,286	65,959	200,245
$30,000$	5%	16,255	4,790	20,999	42,044	134,286	61,170	195,456
$30,000$	10%	16,255	5,539	20,281	42,075	134,286	63,736	198,022

Fig. 2. Results for PostgreSQL

(iii) using the main memory DB, most of the computation time for the IAR-repair is devoted to the detection of minimal conflict sets (i.e., Δ_2^{IAR}); conversely, using the disk-resident DB, a very large percentage of the execution time (always more than 80%) is devoted to the generation of the annotated ABox and to the extraction of the IAR-repair. This is of course due to the fact that such steps require to create and write a large number of records on the disk. On the other hand, RAM size is a bottleneck for the main memory DB (we were not able to process ABoxes with 100,000 assertions).

5 Ongoing and future work

As above observed, most of the execution time of the algorithm *QuAC-IAR* using a disk-resident DB is due to the creation of the annotated ABox (step 1) and to the creation of the IAR-repair (step 5). Thus, avoiding these steps would dramatically improve the efficiency of this algorithm.

To this aim, we observe that both the above steps could be completely skipped if the database schema used for representing the ABox would present an additional attribute for storing annotations in every relation (the usual DB representation of an ABox uses a unary relation for every concept and a binary relation for every role). This corresponds to the idea of directly using an annotated ABox instead of a standard ABox in the system. In this case, the computation of the IAR-repair could only consist of the steps 2, 3 and 4 of the algorithm *QuAC-IAR*. However, the choice of using an annotated ABox instead of an ABox affects query answering, since the queries evaluated on an annotated ABox should be able to filter out the assertions whose annotation is equal to *cons*.

We are currently experimenting whether this choice is actually feasible. Below we present a table showing the evaluation time of four queries of increasing complexity on the ABoxes considered in the previous section (in particular, the ABoxes with 5% inconsistent assertions). We show the cost of both evaluating the query on the IAR-repair (represented as a standard ABox) and directly on the annotated ABox (with the further selection condition on the annotations).

ABox size	query	query ans. on IAR-repair (nsec)	query ans. on annotated ABox (nsec)	difference (msec)	difference (%)
$1,000$	q1	123,577	105,868	-17	-17%
$1,000$	q2	216,740	226,750	10	4%
$1,000$	q3	1,179,561	295,275	-884	-299%
$1,000$	q4	421,161	600,174	179	30%
$3,000$	q1	138,591	229,060	90	39%
$3,000$	q2	210,581	355,716	145	41%
$3,000$	q3	1,348,179	490,842	-857	-175%
$3,000$	q4	507,396	653,685	146	22%
$10,000$	q1	164,384	339,932	175	52%
$10,000$	q2	267,172	499,696	232	47%
$10,000$	q3	1,347,024	592,475	-754	-127%
$10,000$	q4	491,612	664,465	172	26%
$30,000$	q1	199,417	724,521	525	72%
$30,000$	q2	398,448	905,074	506	56%
$30,000$	q3	1,519,493	944,726	-574	-61%
$30,000$	q4	485,067	1,096,021	610	56%

These first experimental results show that, in Quonto, evaluating queries on the annotated ABox often seems computationally not much harder (and sometimes even easier) than evaluating them on the standard ABox. Therefore, a more detailed experimental analysis is needed in order to understand the conditions under which it could be convenient to only work with an annotated representation of the ABox.

Finally, it would be very interesting to compare the performance of QuAC with a query rewriting approach. Indeed, techniques for the perfect rewriting of unions of conjunctive queries over *DL-Lite*$_A$ KBs under both IAR and $ICAR$ semantics have been recently defined [5]. Such techniques are able to reduce query answering over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ to answering a FOL query over the ABox $\mathcal{A}$. So, the ABox is not repaired at all by this approach: rather, the ABox repair is virtually considered during query answering through a suitable reformulation of the query. We plan to implement such query rewriting techniques, with the aim of comparing such an approach with the approach of QuAC.

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
3. J. Dolby, J. Fan, A. Fokoue, A. Kalyanpur, A. Kershenbaum, L. Ma, J. W. Murdock, K. Srinivas, and C. A. Welty. Scalable cleanup of information extraction data using ontologies. In *ISWC/ASWC*, pages 100–113, 2007.
4. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Proc. of RR 2010*, 2010.
5. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Query rewriting for inconsistent DL-Lite ontologies. In *Proc. of RR 2011*, 2011. To appear.
6. R. Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *Proc. of IJCAI 2011*, 2011. To appear.
7. Z. Wang, K. Wang, and R. W. Topor. A new approach to knowledge base revision in *DL-Lite*. In *Proc. of AAAI 2010*. AAAI Press, 2010.