Inconsistency-Tolerant First-Order Rewritability of DL-Lite with Identification and Denial Assertions

Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati,
Marco Ruzzi, and Domenico Fabio Savo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti
Sapienza Università di Roma
lembo,lenzerini,rosati,ruzzi,savo@dis.uniroma1.it

Abstract. This paper is motivated by two requirements arising in practical applications of ontology-based data access (OBDA): the need of inconsistency-tolerant semantics, which allow for dealing with classically inconsistent specifications; and the need of expressing assertions which go beyond the expressive abilities of traditional Description Logics, namely identification and denial assertions. We consider an extension of DL-Lite (the most used DL in OBDA) which allows for the presence of both the aforementioned kinds of assertions in the TBox. We adopt an inconsistency-tolerant semantics for such a DL, specifically the so-called Intersection ABox Repair (IAR) semantics, and we study query answering in this setting. Our main contribution is a query rewriting technique which is able to take into account both identification and denial assertions. By virtue of this technique, we prove that conjunctive query answering under such semantics is first-order rewritable in the considered extension of DL-Lite.

1 Introduction

Ontology-based data access (OBDA) is a computing paradigm which considers the problem of accessing data stored in autonomous databases through the use of an ontology, which provides a conceptual and shared representation of the domain of interest [10]. The main components of an OBDA system are the ontology, the set of data sources to be accessed, and the mapping, which specifies the relationship between the ontology and the data sources. The reasoning service of main interest is query answering, which amounts to process a query expressed in terms of the ontology alphabet, and to compute its answer on the basis of the knowledge specified by the ontology, the mapping, and the data stored at the sources. Various languages for specifying the ontology in this context have been recently proposed [3,2,7], which are designed in order to allow for tractable query answering w.r.t. the size of the data (data complexity). Among these, the members of the *DL-Lite* family of light-weight Description Logics (DLs) present the distinguishing feature of *first-order rewritable query answering* for unions of conjunctive queries (UCQs), which means that such a reasoning service can be reduced to the evaluation of a first-order query (directly translatable into SQL) over a database. This turns out to be a crucial aspect in OBDA, where data are in general not moved from the sources, and are usually managed by a Relational Database Management System.

In the last years we have experimented (see, e.g., [11]) that two important requirements arise in OBDA applications: (i) the need of declarative mechanisms for dealing

with data that are inconsistent with respect to the intensional knowledge specified by the ontology, and (ii) the need of expressing assertions which go beyond the expressive abilities of traditional DLs, namely *identification assertions* and *denial assertions*. Solutions aiming at fulfilling these two requirements should of course still guarantee enough efficiency. Ideally, they should allow for inconsistency-tolerant first-order rewritable query answering of UCQs. Devising one such solution is the goal of the present paper.

Motivated by the first mentioned requirement, we have recently proposed various *inconsistency-tolerant semantics* [8], inspired by the studies on inconsistency handling in belief revision [6] and by the work on consistent query answering in databases [5]. Later works [9,1] have then shown that answering UCQs under the so-called *Intersection ABox Repair (IAR) semantics* is first-order rewritable for ontologies specified in *DL-Lite*$_A$, a member of the *DL-Lite* family [9]. In the present paper we extend the above result, and show that first-order rewritability of conjunctive query answering still holds if we enrich *DL-Lite*$_A$ with identification and denial assertions.

Identification assertions (IdAs) are mechanisms for specifying a set of properties that can be used to identify instances of concepts. IdAs we study here have been originally presented in [4]. Such assertions allow for sophisticated forms of object identification, which may include paths realized through the chaining of roles, their inverses, and attributes. Roughly speaking, when we say that instances of a concept C are identified by a path π, we impose that no two instances of C with overlapping sets of π-fillers exist, i.e., in every interpretation $\mathcal{I}$, no two instances o and o' of C exist with a non-empty intersection of the set of objects reachable by o and by o' in $\mathcal{I}$. This naturally extends to IdAs involving more than one path. Identification assertions are very useful in practice and are essential to represent n-relations and attributes of roles through reification. Indeed, reification is the only way to model n-relations and role attributes in ontology languages, such as OWL, which do not natively allow for such constructs.

Denial Assertions (DAs) are instead used to impose that the answer to a certain boolean conjunctive query over the ontology is false, analogous to negative constraints in [2]. This is particularly useful to specify general forms of disjointness, which is again not supported in traditional ontology languages.

The query rewriting algorithm given in this paper elegantly deals with all forms of inconsistency that can arise in *DL-Lite*$_A$ enriched with IdAs and DAs, thus it represents an alternative to the rewriting algorithm given in [9] for *DL-Lite*$_A$ only. Moreover, it properly manages all the inconsistencies caused by erroneous assignments of values to attributes, i.e., arising when a value of type T_i is assigned to an attribute of value-type T_j disjoint from T_i. This kind of inconsistency has not been considered in [9].

In this paper, for the sake of simplicity, we consider the setting of a single ontology constituted by a TBox and an ABox. However, our technique can be easily extended to the OBDA setting, where mapping assertions relate the TBox to the data sources [10].

The paper is organized as follows. In section 2 we provide some preliminaries. In Section 3 we provide some general properties of the *IAR*-semantics, useful for the rest of the paper. In Section 4, we show first-order rewritability of query answering of UCQs under the *IAR*-semantics. In Section 5 we conclude the paper.

2 Preliminaries

Let Σ be a signature partitioned into Σ_P, containing symbols for predicates, i.e., atomic concepts, atomic roles, attributes and value-domains, and Σ_C, containing symbols for individual (object and value) constants. Given a DL language $\mathcal{L}$, an $\mathcal{L}$-ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ over Σ consists of a TBox $\mathcal{T}$, i.e., a set of intensional assertions over Σ expressed in $\mathcal{L}$, and an ABox $\mathcal{A}$, i.e., a set of extensional assertions over Σ expressed in $\mathcal{L}$. We assume that ABox assertions are always atomic, i.e., they correspond to *ground atoms*, and therefore we omit to refer to $\mathcal{L}$ when we talk about ABox assertions.

The semantics of a DL ontology $\mathcal{O}$ is given in terms of first-order (FOL) interpretations. We denote with $Mod(\mathcal{O})$ the set of models of $\mathcal{O}$, i.e., the set of FOL-interpretations that satisfy all the assertions in $\mathcal{T}$ and $\mathcal{A}$, where the definition of satisfaction depends on the DL language in which $\mathcal{O}$ is specified. An ontology $\mathcal{O}$ is *satisfiable* if $Mod(\mathcal{O}) \neq \emptyset$, and $\mathcal{O}$ *entails* a FOL-sentence ϕ, denoted $\mathcal{O} \models \phi$, if ϕ is satisfied by every $\mathcal{I} \in Mod(\mathcal{O})$. The above definitions naturally apply to a TBox alone.

In the following, we consider DL ontologies $\langle \mathcal{T}, \mathcal{A} \rangle$ in which the TBox $\mathcal{T}$ is always satisfiable, whereas the ABox $\mathcal{A}$ may contradict $\mathcal{T}$. When this happens, we say that $\mathcal{A}$ is $\mathcal{T}$*-inconsistent*, $\mathcal{T}$-consistent otherwise. For ontologies of this kind, the *Intersection ABox Repair (IAR)* semantics introduced in [8] allows for meaningful reasoning in the presence of inconsistency, which is instead trivialized under the FOL-semantics. The *IAR*-semantics is defined in terms of *A*-repairs: given an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, an *A-repair* for $\mathcal{O}$ is a set $\mathcal{A}' \subseteq \mathcal{A}$ such that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$, and there does not exist $\mathcal{A}''$ such that $\mathcal{A}' \subset \mathcal{A}'' \subseteq \mathcal{A}$ and $Mod(\langle \mathcal{T}, \mathcal{A}'' \rangle) \neq \emptyset$. In other terms, an *A*-repair for $\mathcal{O}$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. The set of *A*-repairs for $\mathcal{O}$ is denoted by *AR-Set*$(\mathcal{O})$. The *IAR*-semantics is then given in terms of *IAR*-models, as follows.

Definition 1. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL ontology. The set of Intersection ABox Repair (IAR)-models of $\mathcal{O}$ is defined as* $Mod_{IAR}(\mathcal{O}) = Mod(\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \textit{AR-Set}(\mathcal{O})} \mathcal{A}_i \rangle)$

Notice that if $\mathcal{O}$ is satisfiable under FOL-semantics, then $Mod(\mathcal{O}) = Mod_{IAR}(\mathcal{O})$. The notion of consistent entailment is then the natural extension of classical entailment to *IAR*-semantics: a FOL-sentence ϕ is *IAR-consistently entailed*, or simply *IAR-entailed*, by $\mathcal{O}$, written $\mathcal{O} \models_{\textit{IAR}} \phi$, if $\mathcal{I} \models \phi$ for every $\mathcal{I} \in Mod_{IAR}(\mathcal{O})$.

We focus now on *DL-Lite*$_{A,id}$, a DL of the *DL-Lite* family [3] equipped with identification assertions [4]. Since *DL-Lite*$_{A,id}$ distinguishes between object and value constants, we partition the set Σ_C into two disjoint sets Σ_O, which is the set of constants denoting objects, and Σ_V, which is the set of constants denoting values.

Basic *DL-Lite*$_{A,id}$ expressions are defined as follows:

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

A, P, and P^- denote an *atomic concept*, an *atomic role*, and the *inverse of an atomic role*; $\delta(U)$ (resp. $\rho(U)$) denotes the *domain* (resp. the *range*) of an *attribute* U, i.e., the set of objects (resp. values) that U relates to values (resp. objects); $T_1, \ldots, T_n$ are unbounded pairwise disjoint predefined value-domains; B is called *basic concept*.

A *DL-Lite*$_{A,id}$ TBox is a finite set of the following assertions:

$$B \sqsubseteq C \quad Q \sqsubseteq R \quad U \sqsubseteq V \quad E \sqsubseteq F \quad (\mathsf{funct}\ Q) \quad (\mathsf{funct}\ U) \quad (\mathsf{id}\ B\ \pi_1, \ldots, \pi_n)$$

The assertions above, from left to right, respectively denote inclusions between concepts, roles, attributes, and value-domains, global functionality on roles and on attributes, and identification assertions (IdAs). In IdAs, π_i is a *path*, which is an expression built according to the following syntax: $\pi \longrightarrow S \mid D? \mid \pi_1 \circ \pi_2$, where S denotes an atomic role, the inverse of an atomic role, an attribute, or the inverse of an attribute, $\pi_1 \circ \pi_2$ denotes the composition of paths π_1 and π_2, and $D?$, called *test relation*, represents the identity relation on instances of D, which can be a basic concept or a value-domain. Test relations are used to impose that a path involves instances of a certain concept or value-domain. In our logic, IdAs are *local*, i.e., at least one $\pi_i \in \{\pi_1, ..., \pi_n\}$ has length 1, i.e., it is an atomic role, the inverse of an atomic role, or an attribute. Intuitively, an IdA of the above form asserts that for any two different instances o, o' of B, there is at least one π_i such that o and o' differ in the set of their π_i-fillers, that is the set of objects that are reachable from o by means of π_i. For example, the IdA $(\mathsf{id}\ Match\ homeTeam, visitorTeam)$ says that there are not two different matches with the same home team and host team (which is indeed what happens, for instance, in a season schedule of a football league).

Inclusion assertions that do not contain (resp. contain) the symbols '$\neg$' in the right-hand side are called *positive inclusions* (resp. *negative inclusions*). The set of positive (resp., negative) inclusions in $\mathcal{T}$ will be denoted by $\mathcal{T}^+$ (resp., $\mathcal{T}^-$).

A *DL-Lite*$_{A,id}$ ABox is a finite set of assertions of the form $A(a)$, $P(a, b)$, and $U(a, v)$, where A is an atomic concept, P is an atomic role, U is an attribute, a and b are constants of Σ_O, and v is a constant of Σ_V. In a *DL-Lite*$_{A,id}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, the following condition holds: each role or attribute that either is functional in $\mathcal{T}$ or appears (in either direct or inverse direction) in a path of an IdA in $\mathcal{T}$ is not specialized, i.e., it does not appear in the right-hand side of assertions of the form $Q \sqsubseteq Q'$ or $U \sqsubseteq U'$.

The semantics of *DL-Lite*$_{A,id}$ is given in [4]. We only recall here that each value-domain T_i is interpreted by means of the same function, denoted $val(T_i)$, in every interpretation, and each constant $c_i \in \Sigma_V$ is interpreted as one specific value, denoted $val(c_i)$. Also, the Unique Name Assumption is adopted.

A *boolean conjunctive query (BCQ)* over a *DL-Lite*$_A$ ontology is an expression of the form $q = \exists \vec{y}.conj(\vec{t})$ where $\vec{y}$ is a set of variables and $\vec{t}$ is a set of terms (i.e., constants or variables) such that each variable in $\vec{t}$ is also in $\vec{y}$, and $conj(\vec{t})$ is a conjunction of atoms of the form $A(z)$, $T(z')$, $P(z, z')$ $U(z, z')$ where A is an atomic concept, T is a value-domain, P is an atomic role and U is an attribute, and z, z' are terms. In the following, we will mainly consider *boolean unions of conjunctive queries (BUCQs)*, i.e., first order sentences of the form $Q = \bigvee_{i=1}^{n} q_i$ where $q_i = \exists \vec{y_i}.conj_i(\vec{t_i})$ is a BCQ.

A BCQ q is satisfied by an ABox $\mathcal{A}$ (denoted by $\langle \emptyset, \mathcal{A} \rangle \models q$) if there exists a substitution σ from the variables in q to constants of Σ_C such that the formula $\sigma(q)$ is satisfied in the FOL-interpretation structure where the ABox $\mathcal{A}$ determines the extension of concepts, roles and attributes and the function val determines the extension of the value-domains. With a little abuse of notation we sometimes use $\sigma(q)$ to indicate the set of the atoms in $\sigma(q)$. When query q is satisfied by $\mathcal{A}$, we say that $\sigma_{\mathcal{A}}(q) = \sigma(q) \cap \mathcal{A}$

is the *image of* q *in* $\mathcal{A}$. A BUCQ $Q = \bigvee_{i=1}^{n} q_i$ is satisfied by an ABox $\mathcal{A}$ if there exists $i \in \{1, \ldots n\}$ such that q_i is satisfied by $\mathcal{A}$.

All the results of the present work can be straightforwardly extended to *non-boolean* UCQs, by imposing the (natural) condition that every free variable in the query appears in at least one atom not involving value-domains.

With the notion of query in place we introduce now *denial assertions (DAs)*. A DA is an expression of the form $\forall \vec{y}.conj(\vec{t}) \rightarrow \bot$ where $conj(\vec{t})$ is defined as for BCQs. A DA is satisfied by an interpretation $\mathcal{I}$ if the formula $\exists \vec{y}.conj(\vec{t})$ evaluates to `false` in $\mathcal{I}$. DAs are used to impose general forms of disjointness. For example, the denial $\forall x, y.Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, y) \rightarrow \bot$ says that a match cannot have the same team as both home and visitor team.

In the following we will consider *DL-Lite*$_{A,id}$ ontologies whose TBoxes are enriched with DAs, and call them *DL-Lite*$_{A,id,den}$ ontologies.

3 Properties of the *IAR*-Semantics

We discuss here some properties of the *IAR*-semantics that will be used in the rest of the paper. We recall that the *IAR*-semantics is defined for DL ontologies composed by a consistent TBox and an ABox comprising only atomic assertions, therefore we consider below only this kind of ontologies. We start with the notion of inconsistent set.

Definition 2. *Let $\mathcal{T}$ be a TBox and let $\mathcal{A}$ be an ABox. $\mathcal{A}$ is a* minimal $\mathcal{T}$-inconsistent set *if $\mathcal{A}$ is $\mathcal{T}$-inconsistent, i.e., the ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable, and there is no $\mathcal{A}' \subset \mathcal{A}$ such that $\mathcal{A}'$ is $\mathcal{T}$-inconsistent.*

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL ontology. We denote with $minIncSets(\mathcal{O})$ the set of minimal $\mathcal{T}$-inconsistent sets contained in $\mathcal{A}$. Notice that $\mathcal{O}$ is satisfiable iff $minIncSets(\mathcal{O}) = \emptyset$. Then, the following property holds.

Proposition 1. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL ontology such that $minIncSets(\mathcal{O}) \neq \emptyset$, and let α be an ABox assertion in $\mathcal{A}$. An A-repair $\mathcal{A}'$ of $\mathcal{O}$ such that $\alpha \notin \mathcal{A}'$ exists iff there exists $V \in minIncSets(\mathcal{O})$ such that $\alpha \in V$.*

Let now *AR-Set*$(\mathcal{O}) = \{\mathcal{A}_1, \ldots \mathcal{A}_n\}$ be the set of *A*-repairs of an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, and let q be a BCQ. From Definition 1, we derive that $\mathcal{O} \models_{IAR} q$ iff there exists a subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{A}' \subseteq \mathcal{A}_i$ for every $1 \leq i \leq n$ and $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$. From the observation above, we derive the following theorem, which characterizes the notion of *IAR*-entailment of BCQs.

Theorem 1. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL ontology such that $minIncSets(\mathcal{O}) \neq \emptyset$, and let q be a BCQ. $\mathcal{O} \models_{IAR} q$ iff there exists $\mathcal{A}' \subseteq \mathcal{A}$ such that: (i) $\mathcal{A}'$ is $\mathcal{T}$-consistent; (ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$; (iii) $\mathcal{A}' \cap V = \emptyset$ for every $V \in minIncSets(\mathcal{O})$.*

4 Query Rewriting under *IAR*-Semantics in *DL-Lite*$_{A,id,den}$

In this section, we focus on *DL-Lite*$_{A,id,den}$ ontologies, and provide our technique for computing FOL-rewritings of BUCQs under the *IAR*-semantics. The notion of FOL-rewritability is essentially the same used under FOL-semantics [3]. More precisely,

BUCQs in *DL-Lite*$_{A,id,den}$ are FOL-rewritable under the *IAR*-semantics if, for each BUCQs q and each *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, there exists a FOL-query q_r such that, for any ABox $\mathcal{A}$ $\langle \mathcal{T}, \mathcal{A} \rangle, \models_{IAR} q$ iff $\langle \emptyset, \mathcal{A} \rangle \models q_r$. Such q_r is called the *IAR-perfect reformulation* of q w.r.t. $\mathcal{T}$.

To come up with our reformulation method, we exploit Theorem 1. Roughly speaking, in the reformulation of a BUCQ q over a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, we encode into a FOL-formula all violations that can involve assertions belonging to images of q in any ABox $\mathcal{A}$. Indeed, this can be done by reasoning only on the TBox, and considering each query atom separately. Intuitively, we deal with inconsistency by rewriting each atom α of q into a FOL-formula α_r in such a way that $\langle \emptyset, \mathcal{A} \rangle \models \alpha_r$ only if there exists a substitution σ of the variables in q to constants of Σ_C such that q is satisfied by $\mathcal{A}$, and $\sigma(\alpha)$ does not belong to any minimal $\mathcal{T}$-inconsistent set.

This inconsistency-driven rewriting is then suitably casted into the final reformulation, which takes into account also positive knowledge of the TBox, i.e., the inclusion assertions in $\mathcal{T}^+$. As we will show later in this section, this can be done by means of a slight variation of the algorithm PerfectRef of [3,10]. To formalize the above idea, we need to introduce some preliminary definitions.

We consider Boolean queries corresponding to FOL-sentences of the form:

$$\exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} H_i(t_i^1) \wedge \bigwedge_{i=1}^{m} \neg T_i(t_i^2) \wedge \bigwedge_{i=1}^{\ell} S_i(t_i^3, t_i^4) \wedge \bigwedge_{i=1}^{h} t_i^5 \neq t_i^6 \qquad (1)$$

where every H_i is an unary predicate, which is either an atomic concept or a value-domain, every T_i is a value-domain, every S_i is a binary predicate, which is either an atomic role or an attribute, every t_i^j is a term (i.e., either a constant or a variable), and $z_1, \ldots, z_k$ are all the variables of the query. Notice that sentences of the form (1) are BCQs enriched with limited forms of inequalities and negation.

Given a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, we denote with $contr(\mathcal{T})$ the set of all negative inclusions, functionalities, identifications, denials, and value-domain inclusions occurring in $\mathcal{T}$. Intuitively, the set $contr(\mathcal{T})$ contains all those assertions in $\mathcal{T}$ which can be contradicted by assertions in the ABox. Now, to each assertion $\tau \in contr(\mathcal{T})$, we associate a Boolean query, denoted $\varphi(\tau)$, which encodes the violation of τ, i.e., it looks for images of the negation of τ. Below we provide some of such encodings. Missing cases are can be obtained in a similar way.

- $\varphi((\mathsf{funct}\ P))$: $\exists x, x_1, x_2.P(x, x_1) \wedge P(x, x_2) \wedge x_1 \neq x_2$
- $\varphi(A_1 \sqsubseteq \neg A_2)$: $\exists x.A_1(x) \wedge A_2(x)$
- $\varphi(\rho(U) \sqsubseteq T)$: $\exists x_1, x_2.U(x_1, x_2) \wedge \neg T(x_2)$
- $\varphi(\forall \vec{x}.conj(\vec{t}) \rightarrow \bot)$: $\exists \vec{x}.conj(\vec{t})$

For example, $\varphi((\mathsf{id}\ \ Match\ \ homeTeam, visitorTeam)) = \exists x, x', y, z.Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, z) \wedge Match(x') \wedge homeTeam(x', y) \wedge visitorTeam(x', z) \wedge x \neq x'$, and $\varphi(\forall x, y.Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, y) \rightarrow \bot) = \exists x, y.Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, y)$.

Then, the algorithm unsatQueries($\mathcal{T}$) computes the first-order reformulation under FOL-semantics of $\varphi(\tau)$ for each $\tau \in contr(\mathcal{T})$, and returns the union of all such reformulations. To this aim, unsatQueries($\mathcal{T}$) makes use of the algorithm

$\mathsf{PerfectRef}_{\neq}(\mathcal{T}^+, \varphi(\tau))$, which is a slight variation of the algorithm PerfectRef given in [3]. In this modified version, inequality is considered as a primitive role and negated value-domains are considered as primitive concepts, thus inequality and negated atoms are never rewritten by the algorithm, and the algorithm does not unify query atoms if this causes a violation of an inequality. Note that the result of $\mathsf{PerfectRef}_{\neq}$ is a union of boolean queries of the form (1), represented as a set of queries, as usual.

For example, assume to have a TBox containing the above mentioned IdA τ_1 : (id $Match$ $homeTeam, visitorTeam$) and the inclusion assertion $playedMatch \sqsubseteq Match$. The algorithm $\mathsf{PerfectRef}_{\neq}(\mathcal{T}^+, \varphi(\tau_1))$ returns, among others, the query:

$$\exists\, x, x', y, z.\ playedMatch(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, z) \wedge playedMatch(x') \wedge homeTeam(x', y) \wedge visitorTeam(x', z) \wedge x \neq x'$$

Notice that the query $\varphi(\tau_1)$ cannot be rewritten by unifying $Match(x)$ and $Match(x')$ because of the inequality $x \neq x'$. Such an inequality actually causes that in this example the algorithm can never rewrite queries through unification.

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_{A,id,den}$ ontology. Since $\mathcal{O}$ is satisfiable iff there are no $\mathcal{T}$-inconsistent sets in $\mathcal{A}$, the following theorem states that the query produced by the algorithm $\mathsf{unsatQueries}(\mathcal{T})$ can be used to check satisfiability of $\mathcal{O}$.

Theorem 2. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$ ontology. A $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$ exists iff $\langle \emptyset, \mathcal{A} \rangle \models$ unsatQueries$(\mathcal{T})$.*

If $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{unsatQueries}(\mathcal{T})$, then there exists $q \in \mathsf{unsatQueries}(\mathcal{T})$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. This implies that there exists a substitution σ from the variables in q to constants of Σ_C such that $\sigma(q)$ projected over its positive atoms is contained in $\mathcal{A}$. Similar to BCQs, we call this set the image of q in $\mathcal{A}$, denoted $\sigma_{\mathcal{A}}(q)$. It is possible to show that $\sigma_{\mathcal{A}}(q)$ is a $\mathcal{T}$-inconsistent set contained in $\mathcal{A}$ (in fact, this is part of the proof of Theorem 2). In order to exploit Theorem 1 towards the definition of a FOL-rewriting procedure, we need however to identify those assertions in $\mathcal{A}$ that participate to a *minimal* $\mathcal{T}$-inconsistent set. From the definition of minimal $\mathcal{T}$-inconsistent set, and from Theorem 2 it follows that $\sigma_{\mathcal{A}}(q)$ is a minimal $\mathcal{T}$-inconsistent set iff for every $V \subset \sigma_{\mathcal{A}}(q)$ and every query $q' \in \mathsf{unsatQueries}(\mathcal{T})$, we have that $\langle \emptyset, V \rangle \not\models q'$.

Based on the above observations, we introduce the algorithm $\mathsf{minUnsatQueries}(\mathcal{T})$ which, starting from the set $\mathsf{unsatQueries}(\mathcal{T})$, computes a new set $\mathcal{Q}_{min}$ of queries, which enjoys the following properties: (i) For each query $q \in \mathcal{Q}_{min}$, $\langle \emptyset, \mathcal{A} \rangle \models q$ iff there exists in $\mathsf{unsatQueries}(\mathcal{T})$ a query q' such that $\langle \emptyset, \mathcal{A} \rangle \models q'$. This guarantees that Theorem 2 also holds with $\mathsf{minUnsatQueries}(\mathcal{T})$ in place of $\mathsf{unsatQueries}(\mathcal{T})$. (ii) For each query $q \in \mathcal{Q}_{min}$, if $\langle \emptyset, \mathcal{A} \rangle \models q$, then for every image $\sigma_{\mathcal{A}}(q)$ of q in $\mathcal{A}$, $\langle \emptyset, V \rangle \not\models q'$, where $V \subset \sigma_{\mathcal{A}}(q)$ and $q' \in \mathcal{Q}_{min}$. This guarantees that if a query $q \in \mathcal{Q}_{min}$ is such that $\langle \emptyset, \mathcal{A} \rangle \models q$, then every image of q in $\mathcal{A}$ is a minimal $\mathcal{T}$-inconsistent set.

Before presenting the algorithm $\mathsf{minUnsatQueries}(\mathcal{T})$, we need to introduce some preliminary notions. Given a query q, we say that a term t occurs in an *object position* of q if q contains an atom of the form $A(t)$, $P(t, t')$, $P(t', t)$, $U(t, t')$, whereas we say that t occurs in a *value position* of q if q contains an atom of the form $T(t)$, $U(t', t)$, where A, P, U, and T have the usual meaning. Given two terms t_1 and t_2 occurring in a query q, we say that t_1 and t_2 are *compatible* in q if either both t_1 and t_2 appear only

in object positions of q or both t_1 and t_2 appear only in value positions of q. Moreover, let q and q' be two boolean queries. We say that q is a *proper syntactical subset* of q', written $q \prec_{Rn} q'$ if there exists a renaming function $Rn(q, q')$ of the variables in q to the variables in q', such that every atom $S(\vec{t})$ occurring in $Rn(q, q')$ occurs also in q' and an analogous renaming from q' to q does not exists. The algorithm minUnsatQueries($\mathcal{T}$) is given below.

Algorithm: minUnsatQueries($\mathcal{T}$)
Input: a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$
Output: a set of queries

```
1  Q' ← unsatQueries(T);
2  Q'' ← saturate(Q');
3  Q''' ← ∅;
4  while Q''' ≠ Q'' do
5  |   Q''' ← Q'';
6  |   foreach q ∈ Q'' do
7  |   |   foreach atom U(t, t') in q do
8  |   |   |   if there exist no atoms T(t') and ¬T(t') in q then
9  |   |   |   |   foreach value-domain T in {T_1, ... T_n} do
10 |   |   |   |   |   Q'' ← Q'' \ {q};
11 |   |   |   |   |   Q'' ← Q'' ∪ {q ∧ T(t')};
12 |   |   |   |   |   Q'' ← Q'' ∪ {q ∧ ¬T(t')};
13 foreach q ∈ Q''' do
14 |   foreach term t occurring in q do
15 |   |   if both T_i(t) and T_j(t) occur in q, with i ≠ j then Q''' ← Q''' \ {q};
16 foreach q and q' in Q''' do
17 |   if q ≺_Rn q' then Q''' ← Q''' \ {q'};
18 return Q'''
```

The algorithm proceeds as follows.

Step 1 (line 1): the algorithm initializes $\mathcal{Q}'$ to the set unsatQueries($\mathcal{T}$).

Step 2 (line 2): the algorithm computes the set $\mathcal{Q}''$ through the algorithm saturate. Starting from each query $q \in \mathcal{Q}'$, such an algorithm first unifies pairs of compatible terms in q in all passible ways; then, for any query q' computed in this way, for each pair of terms t_1 and t_2 occurring in q' that are syntactically different, it adds the inequality atom $t_1 \neq t_2$ to q'; finally it returns the union of $\mathcal{Q}'$ with this new set of queries. Notice that such an operation is sound and complete with respect to the set of queries in $\mathcal{Q}'$: in particular, no answer to the set of queries is lost by this transformation, since, for every pair of *compatible* terms t_1, t_2 which are forced to be not equal in q, there is a query q' in $\mathcal{Q}'$ where the same terms have been unified. For instance, assume that $\mathcal{Q}'$ contains only the query $q : \exists x, y.Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, y)$. Then, the algorithm saturate returns a set constituted by the queries $q_1 : \exists x, y.Match(x) \wedge homeTeam(x, y) \wedge visitorTeam(x, y) \wedge x \neq y$ and $q_2 : \exists x.Match(x) \wedge homeTeam(x, x) \wedge visitorTeam(x, x)$.

Step 3 (lines 3-12): let $\{T_1 \dots T_n\}$ be the set of value-domains. The algorithm computes the set $\mathcal{Q}'''$ by producing from each query $q \in \mathcal{Q}''$ the following queries: for each atom $U(x, y)$, where U is an attribute name, if no atoms $T(y)$ appears in q, where T is a value-domain, then the algorithm builds $2n$ new queries by substituting the atom $U(x, y)$ with either the conjunction of atoms $U(x, y) \wedge T_i(y)$ or $U(x, y) \wedge \neg T_i(y)$, for

each $1 \leq i \leq n$. In this way, every possible combination is computed. It is easy to verify that such a transformation is also sound and complete. Step 3 is motivated by the need to provide a complete comparison in Step 5, as explained below.
Step 4 (lines 13-15): the algorithm removes from $\mathcal{Q}'''$ every query q in which a term t occurs in two atoms of the form $T_1(t)$ and $T_2(t)$ (T_1 and T_2 are disjoint, and therefore for each constant c in Σ_V, $T_1(c) \wedge T_2(c)$ is a contradiction).
Step 5 (lines 16-17): the algorithm removes from the set $\mathcal{Q}'''$ each query q' such that there is in $\mathcal{Q}'''$ a different query q whose atoms form, up to renaming of the variables in q, a proper subset of the atoms appearing in q'. This simplified form of query containment guarantees that for every ABox $\mathcal{A}$ and every query q in $\mathcal{Q}'''$, there does not exist a query q' in $\mathcal{Q}'''$, such that for every image $\sigma(q)$ of q in $\mathcal{A}$, $\langle \emptyset, V \rangle \models q'$ where $V \subset \sigma(q)$.
Step 6 (line 18): the algorithm terminates by returning the set $\mathcal{Q}'''$.

Let us now continue the example given at Step 2. Assume that $\mathcal{Q}'$ contains also the query $q_3 : \exists x.homeTeam(x, x)$ which is associate to the denial assertion $\forall x.homeTeam(x, x) \rightarrow \bot$. Obviously, besides query q_1 and q_2, $\mathcal{Q}''$ contains also q_3, which is natively in a "saturated" form. Step 3 and Step 4 have no effect in this example, since none of the queries in $\mathcal{Q}''$ has atoms with attributes or value-domains as predicate. Therefore, $\mathcal{Q}''' = \mathcal{Q}''$, and Step 5 eliminates query q_2 from $\mathcal{Q}'''$, since $q_3 \prec_{Rn} q_2$. Notice that this step is crucial to ensure that every image of a query in $\mathcal{Q}'''$ in any ABox $\mathcal{A}$ is a *minimal* $\mathcal{T}$-inconsistent set. Indeed, let us, for instance, pose $\mathcal{A} = \{Match(a), homeTeam(a, a), visitorTeam(a, a)\}$; q_2 has an image in $\mathcal{A}$, which is $\mathcal{A}$ itself, which is also a $\mathcal{T}$-inconsistent set, but not minimal: indeed, the set $\{homeTeam(a, a)\} \subset \mathcal{A}$ is a also $\mathcal{T}$-inconsistent set, since it violates the DA $\forall x.homeTeam(x, x) \rightarrow \bot$. This set is also minimal, and is indeed an image in $\mathcal{A}$ of the query q_3.

The following lemma states that the algorithm minUnsatQueries($\mathcal{T}$) can be used to check if a *DL-Lite*$_{A,id,den}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent.

Lemma 1. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$ ontology. Then, $\langle \emptyset, \mathcal{A} \rangle \models$ minUnsatQueries($\mathcal{T}$) iff $\langle \emptyset, \mathcal{A} \rangle \models$ unsatQueries($\mathcal{T}$).*

The following crucial lemma guarantees instead that one can use the queries produced by the algorithm minUnsatQueries($\mathcal{T}$) in order to compute every minimal $\mathcal{T}$-inconsistent set in $\mathcal{A}$.

Lemma 2. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$ ontology, and let q be a query in minUnsatQueries($\mathcal{T}$). If $\langle \emptyset, \mathcal{A} \rangle \models q$, then every image of q in $\mathcal{A}$ is a minimal $\mathcal{T}$-inconsistent set.*

Let α, β be two atoms. We say that β is compatible with α if there exists a mapping μ of the variables occurring in β to the terms occurring in α such that $\mu(\beta) = \alpha$ (and in this case we denote the above mapping μ with the symbol $\mu_{\alpha/\beta}$). Given an atom α and a query q, we denote by $CompSet(\alpha, q)$ the set of atoms of q which are compatible with α. Then, let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox and let α be an atom, we define $MinIncSet^{\mathcal{T}}(\alpha)$ as follows.

$$MinIncSet^{\mathcal{T}}(\alpha) = \bigvee_{q \in \text{minUnsatQueries}(\mathcal{T}) \wedge CompSet(\alpha,q) \neq \emptyset} \left(\bigvee_{\beta \in CompSet(\alpha,q)} \mu_{\alpha/\beta}(q) \right)$$

The following key property holds.

Theorem 3. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$ ontology, and let α be an ABox assertion. There exists a minimal $\mathcal{T}$-inconsistent set V in $\mathcal{A}$ such that $\alpha \in V$ iff $\langle \emptyset, \mathcal{A} \rangle \models \mathit{MinIncSet}^{\mathcal{T}}(\alpha)$.*

Let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox and let q be a BCQ of the form

$$\exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} A_i(t_i^1) \wedge \bigwedge_{i=1}^{m} P_i(t_i^2, t_i^3) \wedge \bigwedge_{i=1}^{\ell} U_i(t_i^4, t_i^5) \tag{2}$$

where all symbols have the usual meaning. We denote by $\mathit{IncRewr}_{IAR}(q, \mathcal{T})$ the following FOL-sentence:

$$\mathit{IncRewr}_{IAR}(q, \mathcal{T}) = \exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} A_i(t_i^1) \wedge \neg \mathit{MinIncSet}^{\mathcal{T}}(A_i(t_i^1)) \wedge$$

$$\bigwedge_{i=1}^{m} P_i(t_i^2, t_i^3) \wedge \neg \mathit{MinIncSet}^{\mathcal{T}}(P_i(t_i^2, t_i^3)) \wedge \bigwedge_{i=1}^{\ell} U_i(t_i^4, t_i^5) \wedge \neg \mathit{MinIncSet}^{\mathcal{T}}(U_i(t_i^4, t_i^5))$$

Let $\mathcal{Q}$ be a set of CQs, we define $\mathit{IncRewrUCQ}_{IAR}(\mathcal{Q}, \mathcal{T}) = \bigvee_{q_i \in \mathcal{Q}} \mathit{IncRewr}_{IAR}(q_i, \mathcal{T})$. We are then able to give our final results on reformulation of UCQs under the *IAR*-semantics.

Theorem 4. *Let $\mathcal{T}$ be a DL-Lite$_{A,id,den}$ TBox and let Q be a UCQ. For every ABox $\mathcal{A}$, $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} Q$ iff $\langle \emptyset, \mathcal{A} \rangle \models \mathit{IncRewrUCQ}_{IAR}(\mathsf{saturate}(\mathsf{PerfectRef}(\mathcal{T}^+, Q)), \mathcal{T})$.*

In the above theorem, PerfectRef coincides with the algorithm of [10]. This algorithm takes as input the set of positive inclusions $\mathcal{T}^+$ and the query Q, and computes the perfect reformulation under FOL-semantics of Q w.r.t. $\mathcal{T}^+$. PerfectRef$(\mathcal{T}^+, Q)$ returns a set of CQs specified over $\mathcal{T}$. Through this reformulation we first preprocess each query according to "positive" knowledge of the TBox, and then manage it to deal with possible inconsistency. Then, the algorithm saturate previously described is applied to the query thus obtained: this step is necessary for technical reasons (roughly speaking, it is needed in order to exactly identify, for every query atom α, the queries from minUnsatQueries$(\mathcal{T})$ which correspond to inconsistent sets to which an image of α might belong). This query is finally reformulated according to the definition of $\mathit{IncRewrUCQ}_{IAR}$.

The following complexity result is a direct consequence of Theorem 4, since establishing whether $\langle \emptyset, \mathcal{A} \rangle \models \mathit{IncRewrUCQ}_{IAR}(\mathsf{saturate}(\mathsf{PerfectRef}(\mathcal{T}^+, Q)), \mathcal{T})$ simply amounts to evaluating a FOL-query over the ABox $\mathcal{A}$, which is in AC^0 in data complexity.

Corollary 1. *Let $\mathcal{O}$ be a DL-Lite$_{A,id,den}$ ontology and let Q be a UCQ. Deciding whether $\mathcal{O} \models_{IAR} Q$ is in AC^0 in data complexity.*

We finally notice that the above results directly imply that query answering of UCQs in *DL-Lite*$_{A,id,den}$ under standard semantics is FOL-rewritable, and therefore in AC^0 in data complexity, i.e., it has the same computational behavior of all DLs of the *DL-Lite* family.

5 Conclusion

Motivated by the requirements arising in applications of ontology-based data access, we have presented a new algorithm for inconsistency-tolerant query answering in a DL obtained by extending *DL-Lite*$_A$ with identification and denial assertions. The algorithm is based on a rewriting technique, and shows that query answering under the considered inconsistency-tolerant semantics in *DL-Lite* remains first-order rewritable, even when identification and denial assertions are added to the TBox. We will soon experiment our new technique in the context of several ongoing OBDA projects. Our main goal is to devise optimization techniques that allow for simplifying the rewritten query as much as possible, so that the performance of the query answering process remains acceptable even under the inconsistency-tolerant semantics.

References

1. Bienvenu, M.: First-order expressibility results for queries over inconsistent *DL-Lite* knowledge bases. In: Proc. of DL 2011. CEUR, ceur-ws.org, vol. 745 (2011)
2. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of PODS 2009. pp. 77–86 (2009)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning 39(3), 385–429 (2007)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Path-based identification constraints in description logics. In: Proc. of KR 2008. pp. 231–241 (2008)
5. Chomicki, J.: Consistent query answering: Five easy pieces. In: Proc. of ICDT 2007. LNCS, vol. 4353, pp. 1–17. Springer (2007)
6. Gärdenfors, P., Rott, H.: Belief revision. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 4, pp. 35–132. Oxford University Press (1995)
7. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in *DL-Lite*. In: Proc. of KR 2010. pp. 247–257 (2010)
8. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Proc. of RR 2010 (2010)
9. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Query rewriting for inconsistent *DL-Lite* ontologies. In: Proc. of RR 2011 (2011)
10. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics X, 133–173 (2008)
11. Savo, D.F., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Romagnoli, V., Ruzzi, M., Stella, G.: MASTRO at work: Experiences on ontology-based data access. In: Proc. of DL 2010. CEUR, ceur-ws.org, vol. 573, pp. 20–31 (2010)