Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family

Diego Calvanese
Faculty of Computer Science
Free University of Bozen-Bolzano
Piazza Domenicani 3
I-39100 Bolzano, Italy
calvanese@inf.unibz.it

Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113
I-00198 Roma, Italy
lastname@dis.uniroma1.it

Abstract. We propose a new family of Description Logics (DLs), called *DL-Lite*, specifically tailored to capture basic ontology languages, while keeping low complexity of reasoning. Reasoning here means not only computing subsumption between concepts, and checking satisfiability of the whole knowledge base, but also answering complex queries (in particular, unions of conjunctive queries) over the instance level (ABox) of the DL knowledge base. We show that, for the DLs of the *DL-Lite* family, the usual DL reasoning tasks are polynomial in the size of the TBox, and query answering is LogSpace in the size of the ABox (i.e., in data complexity). To the best of our knowledge, this is the first result of polynomial time data complexity for query answering over DL knowledge bases. Notably our logics allow for a separation between TBox and ABox reasoning during query evaluation: the part of the process requiring TBox reasoning is independent of the ABox, and the part of the process requiring access to the ABox can be carried out by an SQL engine, thus taking advantage of the query optimization strategies provided by current Data Base Management Systems. Since it can be shown that even slight extensions to the logics of the *DL-Lite* family make query answering at least NLogSpace in data complexity, thus ruling out the possibility of using on-the-shelf relational technology for query processing, we can conclude that the logics of the *DL-Lite* family are the maximal DLs supporting efficient query answering over large amounts of instances.

1. Introduction

One of the most important research directions in Description Logics (DLs) is concerned with the trade-off between expressive power and computational complexity of sound and complete reasoning. Research carried out in the past on this topic has shown that many DLs with

efficient, i.e., worst-case polynomial time, reasoning algorithms lack modeling power required in capturing conceptual models and basic ontology languages, while most DLs with sufficient modeling power suffer from inherently worst-case exponential time behavior of reasoning [5, 6].

Although the requirement of polynomially tractable reasoning might be less stringent when dealing with relatively small ontologies, we believe that the need for efficient reasoning algorithms is of paramount importance when the ontology system is to manage large amount of objects (e.g., from thousands to millions of instances). This is the case of several important applications where the use of ontologies is advocated nowadays. For example, in the Semantic Web, ontologies are often used to describe the relevant concepts of Web repositories, and such repositories may incorporate very large data sets, which constitute the instances of the concepts in the ontology. In such cases, two requirements emerge that are typically overlooked in DLs. First, the number of objects in the knowledge base requires managing instances of concepts (i.e., ABoxes) in secondary storage. Second, significant queries to be posed to the knowledge base are more complex than the simple queries (i.e., concepts and roles) usually considered in DL research. Another interesting context where these requirements are relevant is data integration, where an ontology is used as a conceptual, unified view of data stored in a collection of heterogeneous sources. Again, queries posed to this unified representation are usually more complex than just simple concept and role expressions, and the amount of data accessed through the ontology is likely to be of significant size. Unfortunately, in these contexts, whenever the complexity of reasoning is exponential in the number of instances (as for example in Fact[1], Racer[2], and in [16]), there is little hope for effective instance management and query answering algorithms.

In this paper we propose a new family of DLs, called the *DL-Lite* family[3], specifically tailored to capture basic ontology languages, while keeping all reasoning tasks tractable, in particular, with polynomial time complexity with respect to the size of the knowledge base. Notably, *reasoning* here means not only computing subsumption between concepts, and checking satisfiability of the whole knowledge base, but also answering complex queries, i.e., unions of conjunctive queries, over the set of instances maintained in secondary storage.

The specific contributions of the present article are the following:

[1] `http://www.cs.man.ac.uk/~horrocks/FaCT/`

[2] `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`

[3] A first description of some members of the *DL-Lite* family appeared in [13].

1. We define the *DL-Lite* family, and show that the DLs of this family are rich enough to capture significant ontology languages. As usual, a knowledge base expressed in any logic of the *DL-Lite* family is constituted by a TBox and an ABox, where the first component specifies general properties of concepts and roles, whereas the second component specifies the instances of concepts and roles. The basic logic of the family is called *DL-Lite*$_{core}$, and allows for expressing (cyclic) ISA assertions on concepts, disjointness between concepts, role-typing, participation constraints, i.e., assertions stating that all instances of a concept participate to a specified role, and non-participation constraints. The two other logics studied in the paper are called *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$, respectively. The former adds to the core the possibility of expressing functionality restrictions on roles, whereas the latter adds ISA and disjointness assertions between roles. Although at first sight both *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$ appear to be very simple DLs, the kind of modeling constructs in these logics makes them suitable for expressing a variety of representation languages widely adopted in different contexts, such as basic ontology languages, conceptual data models (e.g., Entity-Relationship [17]), and object-oriented formalisms (e.g., basic UML class diagrams[4]).

2. For all DLs of the *DL-Lite* family, we present techniques for the usual reasoning tasks of DLs, such as concept and role subsumption, knowledge base satisfiability, and instance checking. We show that all these tasks are computationally tractable. More precisely, the time complexity of both concept and role subsumption is polynomial in the size of the knowledge base, whereas both knowledge base satisfiability and instance checking are in polynomial time with respect to the size of the knowledge base, and in LogSpace with respect to the size of the ABox only, i.e., in what we can call data complexity [30].

3. For all DLs of the *DL-Lite* family, we present algorithms for answering unions of conjunctive queries posed to knowledge bases expressed in such DLs, and we show that their complexity is polynomial with respect to the size of the whole knowledge base, and in LogSpace with respect to the size of the ABox only. Note that this is one of the few results on answering complex queries (i.e., not corresponding simply to a concept or a role) over a DL knowledge base [16]. In fact, answering conjunctive queries in DLs is a challenging problem, even in the case of *DL-Lite*$_{core}$, where the combination

[4] See `http://www.omg.org/uml/`.

of constructs expressible in the knowledge base does not pose particular difficulties to TBox reasoning. Indeed, in spite of the simplicity of *DL-Lite*$_{core}$ TBoxes, the ability of taking TBox knowledge into account during the process of answering (unions of) conjunctive queries goes beyond the two-variable fragments (with counting) of first-order logic represented by DLs [5]. Finally, we observe that the worst-case complexity of query answering is exponential in the size of the queries, but this is unavoidable, as the complexity of database query evaluation is already exponential. Overall, our complexity results show that, despite the expressive power of the DLs of the *DL-Lite* family, the complexity of query answering in these logics is no worse than traditional query evaluation in relational databases.

A notable consequence of the results presented in this article is that our approach is perfectly suited to representing ABox assertions as relations managed in secondary storage by a Data Base Management System (DBMS). Indeed, our query answering algorithms are based on the idea of expanding the original query into a set of queries that can be directly evaluated by an SQL engine over the ABox, thus taking advantage of well-established query optimization strategies supported by current industrial strength relational technology. Note that this was one of the motivations behind several research works done on CLASSIC in the 80's [7]. Based on this idea, we have developed a reasoning system for *DL-Lite*$_{\mathcal{F}}$, called QuOnto [2], whose main core is the query answering component.

In [14], we present further results on the data complexity of DLs, showing in particular that the DLs of the *DL-Lite* family are essentially the maximal DLs for which conjunctive query answering can be computed in LogSpace, and allowing one to delegate query evaluation to a relational engine. Indeed, even slight extensions to *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$ make query answering (actually already instance checking, i.e., answering atomic queries) at least NLogSpace in data complexity, and thus rule out the possibility of using off-the-shelf relational technology for query processing. In this sense, the *DL-Lite* family includes the first DLs specifically tailored for effective query answering over large amounts of instances.

The paper is organized as follows. The next section defines the *DL-Lite* family and the associated reasoning services, arguing that the DLs of this family are indeed interesting logics for capturing the basic modeling constructs of ontology languages. Section 3 deals with traditional DL reasoning services for the *DL-Lite* family, such as concept and role subsumption, knowledge base satisfiability, and instance checking. In Section 4, we discuss preliminary properties of query answering in

the *DL-Lite* family, while in Sections 5 and 6 we address the problem of query answering for *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$, respectively. In Section 7, we compare our approach with related work on reasoning about ontologies, and, finally, in Section 8 we conclude the paper.

2. The *DL-Lite* Family

Description Logics (DLs) [5] are logics that represent the domain of interest in terms of *concepts*, denoting sets of objects, and *roles*, denoting binary relations between (instances of) concepts. Complex concept and role expressions are constructed starting from a set of atomic concepts and roles by applying suitable constructs. Different DLs allow for different constructs. Properties of concepts and roles can be specified through inclusion assertions, stating that every instance of a concept (resp., role) is also an instance of another concept (resp., role).

In this paper, we focus on a family of DLs, called the *DL-Lite family*. In particular, we deal with three DLs of this family, called *DL-Lite*$_{core}$, *DL-Lite*$_{\mathcal{F}}$, and *DL-Lite*$_{\mathcal{R}}$, respectively.

2.1. *DL-Lite*$_{core}$

The language of *DL-Lite*$_{core}$ is the core language for the whole family. Concepts and roles are formed according to the following syntax:

$$\begin{array}{lll} B & \longrightarrow & A \mid \exists R \\ C & \longrightarrow & B \mid \neg B \end{array} \qquad \begin{array}{lll} R & \longrightarrow & P \mid P^- \\ E & \longrightarrow & R \mid \neg R \end{array}$$

where A denotes an *atomic concept*, P an *atomic role*, and P^- the *inverse* of the atomic role P. B denotes a *basic concept*, i.e., a concept that can be either an atomic concept or a concept of the form $\exists R$, and R denotes a *basic role*, i.e., a role that is either an atomic role or the inverse of an atomic role. Note that $\exists R$ is the standard DL construct of unqualified existential quantification on basic roles. Sometimes we write R^- with the intended meaning that $R^- = P^-$ if $R = P$, and $R^- = P$, if $R = P^-$. Finally, C denotes a *(general) concept*, which can be a basic concept or its negation, whereas E denotes a *(general) role*, which can be a basic role or its negation. Sometimes we write $\neg C$ (resp., $\neg E$) with the intended meaning that $\neg C = \neg A$ if $C = A$ (resp., $\neg E = \neg R$ if $E = R$), and $\neg C = A$, if $C = \neg A$ (resp., $\neg E = R$, if $E = \neg R$).

A DL *knowledge base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ represents the domain of interest in terms of two parts, a TBox $\mathcal{T}$, specifying the intensional knowledge, and an ABox $\mathcal{A}$, specifying extensional knowledge.

A *TBox* is formed by a finite set of *inclusion assertions* of the form

$$B \sqsubseteq C$$

i.e., we allow general concepts to occur on the right-hand side of inclusion assertions, whereas only basic concepts may occur on the left-hand side of inclusion assertions. As we said before, such an inclusion assertion expresses that all instances of concept B are also instances of concept C. We observe that we might include $B_1 \sqcup B_2$ in the constructs for the left-hand side of inclusion assertions (where $\sqcup$ denotes union) and $C_1 \sqcap C_2$ in the constructs for the right-hand side (where $\sqcap$ denotes conjunction). In this way, however, we would not extend the expressive capabilities of the language, since these constructs can be simulated by considering that $B_1 \sqcup B_2 \sqsubseteq C$ is equivalent to the pair of assertions $B_1 \sqsubseteq C$ and $B_2 \sqsubseteq C$, and that $B \sqsubseteq C_1 \sqcap C_2$ is equivalent to $B \sqsubseteq C_1$ and $B \sqsubseteq C_2$. Similarly, we might add $\bot$ (denoting the empty set) to the constructs for the left-hand side and $\top$ (denoting the whole domain) to those for the right-hand side.

An *ABox* is formed by a finite set of *membership assertions* on atomic concepts and on atomic roles, of the form

$$A(a) \qquad\qquad P(a, b)$$

stating respectively that the object denoted by the constant a is an instance of A and that the pair of objects denoted by the pair of constants (a, b) is an instance of the role P.

In this article, we will not consider membership assertions involving general concepts and roles. However, it can be shown that, with respect to the forms of reasoning considered here, a membership assertion $C(a)$ can be simulated in *DL-Lite*$_{core}$ by adding $A \sqsubseteq C$ to the TBox, and $A(a)$ to the ABox, where A is a new atomic concept. Membership assertions of the form $E(a, b)$, can be simulated with the same mechanisms, in those extensions of *DL-Lite*$_{core}$ (see later) that allow for inclusion assertions on roles.

The semantics of a DL is given in terms of interpretations, where an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty *interpretation domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each role R a binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$. In particular, for the constructs of *DL-Lite*$_{core}$ we have:

$$\begin{aligned}
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
P^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
(P^-)^{\mathcal{I}} &= \{(o_2, o_1) \mid (o_1, o_2) \in P^{\mathcal{I}}\} \\
(\exists R)^{\mathcal{I}} &= \{o \mid \exists o' . (o, o') \in R^{\mathcal{I}}\} \\
(\neg B)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}} \\
(\neg R)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}
\end{aligned}$$

An interpretation $\mathcal{I}$ is a *model* of an inclusion assertion $B \sqsubseteq C$, if $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. We extend the notion of model also to inclusion assertions of more general forms with respect to the one allowed in *DL-Lite*$_{core}$. An interpretation $\mathcal{I}$ is a *model* of $C_1 \sqsubseteq C_2$, where C_1, C_2 are general concepts, if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. Similarly, $\mathcal{I}$ is a *model* of $E_1 \sqsubseteq E_2$, where E_1, E_2 are general roles, if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$.

To specify the semantics of membership assertions, we extend the interpretation function to constants, by assigning to each constant a a *distinct* object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Note that this implies that we enforce the *unique name assumption* on constants [5]. An interpretation $\mathcal{I}$ is a model of a membership assertion $A(a)$, (resp., $P(a, b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp., $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$).

Given an (inclusion, or membership) assertion α, and an interpretation $\mathcal{I}$, we denote by $\mathcal{I} \models \alpha$ the fact that $\mathcal{I}$ is a model of α. Given a (finite) set of assertions κ, we denote by $\mathcal{I} \models \kappa$ the fact that $\mathcal{I}$ is a model of every assertion in κ. A *model of a KB* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$. With a little abuse of notation, we also write $\mathcal{I} \models \mathcal{K}$. A KB is *satisfiable* if it has at least one model. A KB $\mathcal{K}$ *logically implies* an assertion α, written $\mathcal{K} \models \alpha$, if all models of $\mathcal{K}$ are also models of α. Similarly, a TBox $\mathcal{T}$ *logically implies* an assertion α, written $\mathcal{T} \models \alpha$, if all models of $\mathcal{T}$ are also models of α.

2.2. *DL-Lite*$_{\mathcal{R}}$ AND *DL-Lite*$_{\mathcal{F}}$

We now present two DLs that extend *DL-Lite*$_{core}$, and that are the subject of our investigation in the following sections.

The first DL that we consider is *DL-Lite*$_{\mathcal{R}}$, which extends *DL-Lite*$_{core}$ with the ability of specifying inclusion assertions between roles of the form

$$R \sqsubseteq E$$

where R and E are defined as above. The semantics of this kind of assertions has been already specified in the previous subsection.

In fact, *DL-Lite*$_{\mathcal{R}}$ might be enhanced with the capability of managing qualified existential quantification on the right-hand side of inclusion assertions on concepts [14]. This construct, however, can be simulated by suitably making use of inclusions between roles and unqualified existential quantification of concepts in inclusions between concepts, and therefore we do not consider it explicitly.

The second extension of *DL-Lite*$_{core}$ we consider is called *DL-Lite*$_{\mathcal{F}}$, which extends *DL-Lite*$_{core}$ with the ability of specifying functionality on roles or on their inverses. Assertions used to this aim are of the form

$$(\mathsf{funct}\ R)$$

where R is defined as above.

An interpretation $\mathcal{I}$ is a model of an assertion (funct R) if the binary relation $R^{\mathcal{I}}$ is a function, i.e., $(o, o_1) \in R^{\mathcal{I}}$ and $(o, o_2) \in R^{\mathcal{I}}$ implies $o_1 = o_2$.

Despite the simplicity of the language and of the assertions allowed, the DLs in the *DL-Lite* family are able to capture the main notions (though not all, obviously) of both ontologies and conceptual modeling formalisms used in databases and software engineering (i.e., Entity-Relationship and UML class diagrams). In particular, *DL-Lite* assertions allow us to specify:

- *ISA*, e.g., stating that concept A_1 is subsumed by concept A_2, using $A_1 \sqsubseteq A_2$;
- *disjointness*, e.g., between concepts A_1 and A_2, using $A_1 \sqsubseteq \neg A_2$;
- *role-typing*, e.g., stating that the first (resp., second) component of the relation P is an instance of A_1 (resp., A_2), using $\exists P \sqsubseteq A_1$ (resp., $\exists P^- \sqsubseteq A_2$);
- *mandatory participation*, e.g., stating that all instances of concept A participate to the relation P as the first (resp., second) component, using $A \sqsubseteq \exists P$ (resp., $A \sqsubseteq \exists P^-$);
- *mandatory non-participation*, using $A \sqsubseteq \neg\exists P$ or $A \sqsubseteq \neg\exists P^-$;
- *functionality restrictions* on roles, using (funct P) or (funct P^-).

Notice that *DL-Lite*$_{\mathcal{F}}$ is a strict subset of OWL[5] (in fact of OWL Lite). Notice that, the latter includes constructs (e.g., some kinds of role restrictions) and forms of inclusion assertions that are not expressible in *DL-Lite*, and that make reasoning (already in OWL Lite) non-tractable in general.

Instead, *DL-Lite*$_{\mathcal{R}}$ can be seen as an extension of (the DL-like part of) the ontology language RDFS[6]. Indeed, *DL-Lite*$_{\mathcal{R}}$ adds to RDFS the ability of expressing disjointness of concepts and roles, and mandatory participation and non-participation.

EXAMPLE 1. Consider the atomic concepts *Professor* and *Student*, the roles *TeachesTo* and *HasTutor*, and the following *DL-Lite*$_{core}$ TBox

[5] `http://www.w3.org/TR/owl-features/`
[6] `http://www.w3.org/RDF/`

$\mathcal{T}$:

$$\begin{aligned} \mathit{Professor} &\sqsubseteq \exists \mathit{TeachesTo} \\ \mathit{Student} &\sqsubseteq \exists \mathit{HasTutor} \\ \exists \mathit{TeachesTo}^- &\sqsubseteq \mathit{Student} \\ \exists \mathit{HasTutor}^- &\sqsubseteq \mathit{Professor} \\ \mathit{Professor} &\sqsubseteq \neg \mathit{Student} \end{aligned}$$

Such a TBox states that professors do teach to students, that students have a tutor, which is also a professor, and that no student is also a professor (and vice-versa).

Notice that in *DL-Lite*$_{\mathcal{R}}$ we could add the assertion

$$\mathit{HasTutor}^- \sqsubseteq \mathit{TeachesTo}$$

stating that a tutor also teaches the student s/he is tutoring.

On the other hand, in *DL-Lite*$_{\mathcal{F}}$ we could add the assertion

$$(\mathsf{funct}\ \mathit{HasTutor})$$

stating that everyone has at most one tutor.

Finally, we show a simple ABox $\mathcal{A}$:

$$\mathit{Student}(\mathsf{John}), \quad \mathit{HasTutor}(\mathsf{John}, \mathsf{Mary}), \quad \mathit{TeachesTo}(\mathsf{Mary}, \mathsf{Bill}).$$

■

We conclude this subsection with a brief discussion on the *finite model property* [5]. We remind the reader that a DL enjoys the finite model property if every satisfiable KB expressed in this logic admits a model with a finite domain. It is interesting to observe that, while *DL-Lite*$_{\mathcal{R}}$ has the finite model property [29], *DL-Lite*$_{\mathcal{F}}$ does not, as the following example shows. Consider the *DL-Lite*$_{\mathcal{F}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with

$$\mathcal{T} = \{\ A \sqsubseteq \exists P, \quad \exists P^- \sqsubseteq A, \quad (\mathsf{funct}\ P^-), \quad B \sqsubseteq \exists P \quad B \sqsubseteq \neg A\ \}$$

and $\mathcal{A} = \{B(a)\}$. It is easy to see that $\mathcal{K}$ admits only infinite models.

2.3. Queries

We start with a general notion of queries in first-order logic, and then we move to the definition of queries over a DL KB.

A *query* is an open formula of first-order logic with equalities (FOL in the following). We denote a (FOL) query q as follows

$$\{\, \vec{x} \mid \phi(\vec{x}) \,\}$$

where $\phi(\vec{x})$ is a FOL formula with free variables $\vec{x}$. We call the size of $\vec{x}$ the *arity* of the query q. Given an interpretation $\mathcal{I}$, $q^{\mathcal{I}}$ is the set of tuples of domain elements that, when assigned to the free variables, make the formula ϕ true in $\mathcal{I}$ [1].

A *boolean query* is a query that does not involve any free variable (i.e., it is a closed formula). Given a boolean query $q = \{ \ \mid \phi\}$, we may denote it simply by ϕ. Given an interpretation $\mathcal{I}$, $\phi^{\mathcal{I}}$ consists of the only *empty tuple*, i.e., the tuple of arity 0, in the case in which ϕ is true in $\mathcal{I}$, whereas $\phi^{\mathcal{I}}$ is obviously empty if ϕ is false in $\mathcal{I}$.

Among the various queries, we are interested in conjunctive queries and union of conjunctive queries. A *conjunctive query (CQ)* q is a query of the form

$$\{ \vec{x} \mid \exists \vec{y}.conj(\vec{x}, \vec{y}) \}$$

where $conj(\vec{x}, \vec{y})$ is a conjunction of atoms and equalities, with free variables $\vec{x}$ and $\vec{y}$. A *union of conjunctive queries (UCQ)* q is a query of the form

$$\{ \vec{x} \mid \bigvee_{i=1,\ldots,n} \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i}) \}$$

where each $conj_i(\vec{x}, \vec{y_i})$ is, as before, a conjunction of atoms and equalities with free variables $\vec{x}$ and $\vec{y_i}$. Obviously, the class of union of conjunctive queries contains the class of conjunctive queries.

Sometimes, we use the standard datalog notation (see e.g., [1]) to denote conjunctive queries and unions of conjunctive queries. Namely, a conjunctive query $q = \{ \vec{x} \mid \exists \vec{y}.conj(\vec{x}, \vec{y}) \}$ is denoted in datalog notation as

$$q(\vec{x'}) \leftarrow conj'(\vec{x'}, \vec{y'})$$

where $conj'(\vec{x'}, \vec{y'})$ is the list of atoms in $conj(\vec{x}, \vec{y})$ obtained after having equated the variables $\vec{x}, \vec{y}$ according to the equalities in $conj(\vec{x}, \vec{y})$. As a result of such equality elimination, we have that $\vec{x'}$ and $\vec{y'}$ can actually contain constants and multiple occurrences of the same variable. We call $q(\vec{x'})$ the head of q, and $conj'(\vec{x'}, \vec{y'})$ the body. Moreover, we call the variables in $\vec{x'}$ the *distinguished variables* of q and those in $\vec{y'}$ the *non-distinguished variables*.

The datalog notation is then extended to unions of conjunctive queries as follows. A union of conjunctive queries

$$q = \{ \vec{x} \mid \bigvee_{i=1,\ldots,n} \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i}) \}$$

is denoted in datalog notation as

$$q = \{ q_1, \ldots, q_n \}$$

where each q_i is the datalog expression corresponding to the conjunctive query $q_i = \{ \vec{x} \mid \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i}) \}$.

With the general notion of query in place, we can now define queries over a DL KB. In particular, we will concentrate on conjunctive queries and unions of conjunctive queries. A *conjunctive query over a KB* $\mathcal{K}$ is a conjunctive query whose atoms are of the form $A(z)$ or $P(z_1, z_2)$, where A and P are respectively an atomic concept and an atomic role of $\mathcal{K}$, and z, z_1, z_2 are either constants in $\mathcal{K}$ or variables. Similarly, we define *unions of conjunctive queries over a KB* $\mathcal{K}$.

Given a query q (either a conjunctive query or a union of conjunctive queries) and a KB $\mathcal{K}$, the *answer to* q *over* $\mathcal{K}$ is the set $ans(q, \mathcal{K})$ of tuples $\vec{a}$ of constants appearing in $\mathcal{K}$ such that $\vec{a}^{\mathcal{M}} \in q^{\mathcal{M}}$, for every model $\mathcal{M}$ of $\mathcal{K}$. Notice that by definition $ans(q, \mathcal{K})$ is finite since $\mathcal{K}$ is finite, and hence the number of constants appearing in $\mathcal{K}$ is finite. Notice also that the tuple $\vec{a}$ can be the empty tuple in the case in which q is a boolean conjunctive query. More precisely, in this case the set $ans(q, \mathcal{K})$ consists of the only empty tuple if and only if the formula q is true in every model of $\mathcal{K}$.

Observe that, if $\mathcal{K}$ is unsatisfiable, then $ans(q, \mathcal{K})$ is trivially the set of all possible tuples of constants in $\mathcal{K}$ whose arity is the one of the query. We denote such a set by $AllTup(q, \mathcal{K})$.

2.4. Reasoning services

In studying the *DL-Lite* family, we are interested in several reasoning services. Obviously, we want to take into account traditional DL reasoning services, and in particular, for both *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$ KBs, we consider the following problems:

- *knowledge base satisfiability*, i.e., given a KB $\mathcal{K}$, verify whether $\mathcal{K}$ admits at least one model;
- *logical implication of KB assertions*, which consists of the following subproblems:
  - *instance checking*, i.e., given a KB $\mathcal{K}$, a concept C and a constant a (resp., a role E and a pair of constants a and b), verify whether $\mathcal{K} \models C(a)$ (resp., $\mathcal{K} \models E(a, b)$);
  - *subsumption of concepts or roles*, i.e., given a TBox $\mathcal{T}$ and two general concepts C_1 and C_2 (resp., two general roles E_1 and E_2), verify whether $\mathcal{T} \models C_1 \sqsubseteq C_2$ (resp., $\mathcal{T} \models E_1 \sqsubseteq E_2$).
  - *checking functionality*, i.e., given a TBox $\mathcal{T}$ and a basic role R, verify whether $\mathcal{T} \models (\mathsf{funct}\ R)$.

In addition we are interested in:

- *query answering*, i.e., given a KB $\mathcal{K}$ and a query q (either a conjunctive query or a union of conjunctive queries) over $\mathcal{K}$, compute the set $ans(q, \mathcal{K})$.

The following decision problem, called *recognition problem*, is associated to the query answering problem: given a KB $\mathcal{K}$, a query q (either a conjunctive query or a union of conjunctive queries), and a tuple of constants $\vec{a}$ of $\mathcal{K}$, check whether $\vec{a} \in ans(q, \mathcal{K})$. When we talk about the computational complexity of query answering, in fact we implicitly refer to the associated recognition problem.

In analyzing the computational complexity of a reasoning problem over a DL KB, we distinguish between data complexity and combined complexity [30]: *data complexity* is the complexity with respect to the size of the ABox only, while *combined complexity* is the complexity with respect to the size of all inputs to the problem.

2.5. The notion of FOL-reducibility

We now introduce the notion of FOL-reducibility for both satisfiability and query answering, which will be used in the sequel.

First, given an ABox $\mathcal{A}$ (of the kind considered above), we denote by $db(\mathcal{A}) = \langle \Delta^{db(\mathcal{A})}, \cdot^{db(\mathcal{A})} \rangle$ *the interpretation* defined as follows:

- $\Delta^{db(\mathcal{A})}$ is the non-empty set consisting of all constants occurring in $\mathcal{A}$,
- $a^{db(\mathcal{A})} = a$, for each constant a,
- $A^{db(\mathcal{A})} = \{a \mid A(a) \in \mathcal{A}\}$, for each atomic concept A, and
- $P^{db(\mathcal{A})} = \{(a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A}\}$, for each atomic role P.

Observe that the interpretation $db(\mathcal{A})$ is a minimal model of the ABox $\mathcal{A}$.

Intuitively, FOL-reducibility of satisfiability (resp., query answering) captures the property that we can reduce satisfiability checking (resp., query answering) to evaluating a FOL query over the ABox $\mathcal{A}$ considered as a relational database, i.e., over $db(\mathcal{A})$. The definitions follow.

DEFINITION 2. Satisfiability in a DL $\mathcal{L}$ is *FOL-reducible*, if for every TBox $\mathcal{T}$ expressed in $\mathcal{L}$, there exists a boolean FOL query q, over the alphabet of $\mathcal{T}$, such that for every non-empty ABox $\mathcal{A}$, $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable if and only if q evaluates to false in $db(\mathcal{A})$.

DEFINITION 3. Query answering in a DL $\mathcal{L}$ for unions of conjunctive queries is *FOL-reducible*, if for every union of conjunctive queries q and every TBox $\mathcal{T}$ expressed in $\mathcal{L}$, there exists a FOL query q_1, over the alphabet of $\mathcal{T}$, such that for every non-empty ABox $\mathcal{A}$ and every tuple of constants $\vec{a}$ occurring in $\mathcal{A}$, $\vec{a} \in ans(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ if and only if $\vec{a}^{db(\mathcal{A})} \in q_1^{db(\mathcal{A})}$.

3. Techniques for DL reasoning

In this section, we study traditional DL reasoning services for KBs expressed using the *DL-Lite* family. In particular, we consider knowledge base satisfiability and logical implication, which means (concept/role) instance checking, (concept/role) subsumption, and implication of functionality assertions. We show that all such reasoning services are in PTIME w.r.t. combined complexity, and that instance checking and satisfiability (which make use of the ABox) are FOL-reducible, and hence in LogSpace with respect to data complexity. We start our investigation from KB satisfiability, and then we tackle logical implication. In fact, logical implication can be basically reduced to KB satisfiability, and therefore we first give algorithms for KB satisfiability and then we show how to reduce logical implication to such a service. Finally, we provide the complexity results mentioned above.

Hereinafter, we call *positive inclusions (PIs)* assertions of the form $B_1 \sqsubseteq B_2$ or of the form $R_1 \sqsubseteq R_2$, whereas we call *negative inclusions (NIs)* assertions of the form $B_1 \sqsubseteq \neg B_2$ or $R_1 \sqsubseteq \neg R_2$.

3.1. Knowledge base satisfiability

Our goal in this subsection is to show that knowledge base satisfiability is FOL-reducible. To this aim, we resort to two main constructions, namely the canonical interpretation and the closure of the negative inclusions. We present them in turn below.

3.1.1. *Canonical Interpretation*

The canonical interpretation of a KB expressed either in *DL-Lite*$_{\mathcal{R}}$ or in *DL-Lite*$_{\mathcal{F}}$ is an interpretation constructed according to the notion of *chase* [1]. In particular, we adapt here the notion of *restricted chase* adopted by Johnson and Klug in [25].

We start by defining the notion of applicable positive inclusion assertions (PIs), and then we exploit applicable PIs to construct the chase for *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$ knowledge bases. Finally, with the notion of chase in place, we give the definition of canonical interpretation.

In the following, as usual, we denote an atomic concept with the symbol A, possibly with subscript, an atomic role with the symbol P, possibly with subscript, and a basic role with the symbol R, possibly with subscript. Furthermore, for easyness of exposition, we make use of the function ga that takes as input a basic role and two constants and returns a membership assertion, as specified below

$$\mathsf{ga}(R, a, b) = \begin{cases} P(a, b), & \text{if } R = P \\ P(b, a), & \text{if } R = P^- \end{cases}$$

DEFINITION 4. Let $\mathcal{S}$ be a set of *DL-Lite*$_\mathcal{R}$ or *DL-Lite*$_\mathcal{F}$ membership assertions, and let $\mathcal{T}_p$ be a set of *DL-Lite*$_\mathcal{R}$ or *DL-Lite*$_\mathcal{F}$ PIs. Then, a PI $\alpha \in \mathcal{T}_p$ is *applicable in* $\mathcal{S}$ *to a membership assertion* $f \in \mathcal{S}$ if

- $\alpha = A_1 \sqsubseteq A_2$, $f = A_1(a)$, and $A_2(a) \notin \mathcal{S}$;
- $\alpha = A \sqsubseteq \exists R$, $f = A(a)$, and there does not exist any constant b such that $\mathsf{ga}(R, a, b) \in \mathcal{S}$;
- $\alpha = \exists R \sqsubseteq A$, $f = \mathsf{ga}(R, a, b)$, and $A(a) \notin \mathcal{S}$;
- $\alpha = \exists R_1 \sqsubseteq \exists R_2$, $f = \mathsf{ga}(R_1, a, b)$, and there does not exist any constant c such that $\mathsf{ga}(R_2, a, c) \in \mathcal{S}$;
- $\alpha = R_1 \sqsubseteq R_2$, $f = \mathsf{ga}(R_1, a, b)$, and $\mathsf{ga}(R_2, a, b) \notin \mathcal{S}$.

Applicable PIs can be used, i.e., *applied*, in order to construct the chase of a KB. Roughly speaking, the chase of a *DL-Lite*$_\mathcal{R}$ or *DL-Lite*$_\mathcal{F}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a (possibly infinite) set of membership assertions, constructed step-by-step starting from the ABox $\mathcal{A}$. At each step of the construction, a PI $\alpha \in \mathcal{T}$ is applied to a membership assertion f belonging to the current set $\mathcal{S}$ of membership assertions. Applying a PI means adding a new suitable membership assertion to $\mathcal{S}$, thus obtaining a new set $\mathcal{S}'$ in which α is not applicable to f anymore. For example, if $\alpha = A_1 \sqsubseteq A_2$ is applicable in $\mathcal{S}$ to $f = A_1(a)$, the membership assertion to be added to $\mathcal{S}$ is $A_2(a)$, i.e., $\mathcal{S}' = \mathcal{S} \cup A_2(a)$. In some cases (i.e., $\alpha = A \sqsubseteq \exists R$ or $\alpha = \exists R_1 \sqsubseteq \exists R_2$), to achieve an analogous aim, the new membership assertion has to make use of a new constant symbol that does not occur in $\mathcal{S}$.

Notice that such a construction process strongly depends on the order in which we select both the PI to be applied at each step and the membership assertion to which such a PI is applied, as well as on which constants we introduce at each step. Therefore, a number of syntactically distinct sets of membership assertions might result from this process. However, it is possible to show that the result is unique

up to renaming of constants occurring in each such a set. Since we want our construction process to come out with a unique chase of a certain knowledge base, along the lines of [25], we assume in the following to have a fixed infinite set of constants, whose symbols are ordered in lexicographic way, and we select PIs, membership assertions and constant symbols in lexicographic order. More precisely, given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we denote with Γ_A the set of all constant symbols occurring in $\mathcal{A}$. Also, we assume to have an infinite set Γ_N of constant symbols not occurring in $\mathcal{A}$, such that the set $\Gamma_C = \Gamma_A \cup \Gamma_N$ is totally ordered in lexicographic way. Then, our notion of chase is precisely given below.

DEFINITION 5. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_\mathcal{R}$ or a *DL-Lite*$_\mathcal{F}$ KB, let $\mathcal{T}_p$ be the set of positive inclusion assertions in $\mathcal{T}$, let n be the number of membership assertions in $\mathcal{A}$, and let Γ_N be the set of constants defined above. Assume that the membership assertions in $\mathcal{A}$ are numbered from 1 to n following their lexicographic order, and consider the following definition

- $\mathcal{S}_0 = \mathcal{A}$
- $\mathcal{S}_{j+1} = \mathcal{S}_j \cup \{f_{new}\}$, where f_{new} is a membership assertion numbered with $n + j + 1$ in $\mathcal{S}_{j+1}$ and obtained as follows

let f be the first membership assertion in $\mathcal{S}_j$ such that
  there exists a PI $\alpha \in \mathcal{T}_p$ applicable in $\mathcal{S}_j$ to f
let α be the lexicographically first PI applicable in $\mathcal{S}_j$ to f
let a_{new} be the constant of Γ_N that follows lexicographically
  all constants occurring in $\mathcal{S}_j$
case α, f of
  (cr1) $\alpha = A_1 \sqsubseteq A_2$, $f = A_1(a)$
    then $f_{new} = A_2(a)$
  (cr2) $\alpha = A \sqsubseteq \exists R$ and $f = A(a)$
    then $f_{new} = \mathsf{ga}(R, a, a_{new})$
  (cr3) $\alpha = \exists R \sqsubseteq A$ and $f = \mathsf{ga}(R, a, b)$
    then $f_{new} = A(a)$
  (cr4) $\alpha = \exists R_1 \sqsubseteq \exists R_2$ and $f = \mathsf{ga}(R_1, a, b)$
    then $f_{new} = \mathsf{ga}(R_2, a, a_{new})$
  (cr5) $\alpha = R_1 \sqsubseteq R_2$ and $f = \mathsf{ga}(R_1, a, b)$
    then $f_{new} = \mathsf{ga}(R_2, a, b)$.

Then, we call *chase of* $\mathcal{K}$, denoted *chase*($\mathcal{K}$), the set of membership assertions obtained as the infinite union of all $\mathcal{S}_j$, i.e.,

$$chase(\mathcal{K}) = \bigcup_{j \in \mathbb{N}} \mathcal{S}_j.$$

In the above definition, **cr1**, **cr2**, **cr3**, **cr4**, and **cr5** indicate the five rules that are used for constructing the chase, each one corresponding to the application of a PI. Such rules are called *chase rules*, and we say that a chase rule is applied to a membership assertion f if the corresponding PI is applied to f. Notice that rules **cr1**, **cr2**, **cr3**, **cr4** are applied in the construction of the chase of both *DL-Lite*$_\mathcal{R}$ and *DL-Lite*$_\mathcal{F}$ KBs, whereas **cr5** is meaningful only for *DL-Lite*$_\mathcal{R}$ KBs, since PIs of the form $R_1 \sqsubseteq R_2$ do not occur in *DL-Lite*$_\mathcal{F}$ KBs. Observe also that NIs and functionality assertions in $\mathcal{K}$ have no role in constructing $chase(\mathcal{K})$. Indeed $chase(\mathcal{K})$ depends only on the ABox $\mathcal{A}$ and the PIs in $\mathcal{T}$.

In the following, we will denote with $chase_i(\mathcal{K})$ the portion of the chase obtained after i applications of the chase rules, selected according to the ordering established in Definition 5, i.e., $chase_i(\mathcal{K}) = \bigcup_{j\in\{0,..,i\}} \mathcal{S}_j = \mathcal{S}_i$.

The following property shows that the notion of chase of a knowledge base is fair.

PROPOSITION 6. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ be a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ KB, and let α be a PI in $\mathcal{T}$. Then, if there is an $i \in \mathbb{N}$ such that α is applicable in $chase_i(\mathcal{K})$ to a membership assertion $f \in chase_i(\mathcal{K})$, then there is a $j \geq i$ such that $chase_{j+1}(\mathcal{K}) = chase_j(\mathcal{K}) \cup f'$, where f' is the result of applying α to f in $chase_j(\mathcal{K})$.*

Proof. Assume by contradiction that there is no $j \geq i$ such that $chase_{j+1}(\mathcal{K}) = chase_j(\mathcal{K}) \cup f'$. This would mean that either there are infinitely many membership assertions that precede f in the ordering that we choose for membership assertions in $chase(\mathcal{K})$, or that there are infinitely many chase rules applied to some membership assertion that precedes f. However, none of these cases is possible. Indeed, f is assigned with an ordering number m such that exactly $m-1$ membership assertions precede f. Furthermore, a PI can be applied at most once to a membership assertion (afterwards, the precondition is not satisfied and the PI is not applicable anymore), and also there exists only a finite number ℓ of PIs. Therefore, it is possible to apply a chase rule to some membership assertion at most ℓ times. We can thus conclude that the claim holds. □

With the notion of chase in place we can introduce the notion of canonical interpretation. We define the *canonical interpretation* $can(\mathcal{K})$ as the interpretation $\langle \Delta^{can(\mathcal{K})}, \cdot^{can(\mathcal{K})}\rangle$, where:

- $\Delta^{can(\mathcal{K})} = \Gamma_C$,
- $a^{can(\mathcal{K})} = a$, for each constant a occurring in $chase(\mathcal{K})$,

- $A^{can(\mathcal{K})} = \{a \mid A(a) \in chase(\mathcal{K})\}$, for each atomic concept A, and
- $P^{can(\mathcal{K})} = \{(a_1, a_2) \mid P(a_1, a_2) \in chase(\mathcal{K})\}$, for each atomic role P.

We also define $can_i(\mathcal{K}) = \langle \Delta^{can(\mathcal{K})}, \cdot^{can_i(\mathcal{K})} \rangle$, where $\cdot^{can_i(\mathcal{K})}$ is analogous to $\cdot^{can(\mathcal{K})}$ but it refers to $chase_i(\mathcal{K})$ instead of $chase(\mathcal{K})$. According to the above definition, it is easy to see that $can(\mathcal{K})$ (resp., $can_i(\mathcal{K})$) is unique. Notice also that $can_0(\mathcal{K})$ is tightly related to the interpretation $db(\mathcal{A})$. Indeed, while $\Delta^{db(\mathcal{A})} \subseteq \Delta^{can(\mathcal{K})}$, we have that $\cdot^{db(\mathcal{A})} = \cdot^{can_0(\mathcal{K})}$.

We point out that $chase(\mathcal{K})$ and $can(\mathcal{K})$ (resp., $chase_i(\mathcal{K})$) and $can_i(\mathcal{K})$) are strongly connected. In particular, we note that, whereas $chase_{i+1}(\mathcal{K})$ is obtained by adding a membership assertion to $chase_i(\mathcal{K})$, $can_{i+1}(\mathcal{K})$ can be seen as obtained from $can_i(\mathcal{K})$ by adding either an object to the extension of an atomic concept of $\mathcal{K}$, or a pair of objects to the extension of an atomic role of $\mathcal{K}$ (notice that the domain of interpretation is the same in each $can_i(\mathcal{K})$, and in particular in $can(\mathcal{K})$). By virtue of the strong connection discussed above, in the following we will often prove properties of $can(\mathcal{K})$ (resp., $can_i(\mathcal{K})$) by reasoning over the structure of $chase(\mathcal{K})$ (resp., $chase_i(\mathcal{K})$).

Now, we are ready to show a notable property that holds for $can(\mathcal{K})$.

LEMMA 7. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ KB and let $\mathcal{T}_p$ be the set of positive inclusion assertions in $\mathcal{T}$. Then, $can(\mathcal{K})$ is a model of $\langle \mathcal{T}_p, \mathcal{A} \rangle$.*

Proof. Since $\langle \mathcal{T}_p, \mathcal{A} \rangle$ does not contain NIs, to prove the claim we only need to show that $can(\mathcal{K})$ satisfies all membership assertions in $\mathcal{A}$ and all PIs in $\mathcal{T}_p$. The fact that $can(\mathcal{K})$ satisfies all membership assertions in $\mathcal{A}$ follows from the fact that $\mathcal{A} \subseteq chase(\mathcal{K})$. Then, it remains to prove that $can(\mathcal{K}) \models \mathcal{T}_p$. Let us proceed by contradiction considering all possible cases.

Suppose by contradiction that a PI of the form $A_1 \sqsubseteq A_2 \in \mathcal{T}_p$, where A_1 and A_2 are atomic concepts, is not satisfied by $can(\mathcal{K})$. This means that there exists a constant $a \in \Gamma_C$ such that $A_1(a) \in chase(\mathcal{K})$ and $A_2(a) \notin chase(\mathcal{K})$. However, such a situation would trigger the chase rule **cr1**, since $A_1 \sqsubseteq A_2$ would be applicable to $A_1(a)$ in $chase(\mathcal{K})$ and Proposition 6 ensures that such a PI would be applied at some step in the construction of the chase, thus causing the insertion of $A_2(a)$ in $chase(\mathcal{K})$, hence contradicting the assumption.

Now, assume by contradiction that a PI of the form $A \sqsubseteq \exists R \in \mathcal{T}_p$, where A is an atomic concept and R is a basic role, is not satisfied by $can(\mathcal{K})$. This means that there exists a constant $a \in \Gamma_C$ such that $A(a) \in chase(\mathcal{K})$ and there does not exist a constant $b \in \Gamma_C$ such

that $\mathsf{ga}(R, a, b) \in \mathit{chase}(\mathcal{K})$. However, such a situation would trigger the chase rule **cr2**, since $A \sqsubseteq \exists R$ would be applicable to $A(a)$ in $\mathit{chase}(\mathcal{K})$ and Proposition 6 ensures that such a PI would be applied at some step in the construction of the chase, thus causing the insertion of $\mathsf{ga}(R, a, c)$ in $\mathit{chase}(\mathcal{K})$, where $c \in \Gamma_C$ follows lexicographically all constants occurring in $\mathit{chase}(\mathcal{K})$ before the execution of **cr2**, hence contradicting the assumption.

Then, assume by contradiction that a PI of the form $\exists R \sqsubseteq A \in \mathcal{T}_p$, where R is a basic role and A is an atomic concept, is not satisfied by $\mathit{can}(\mathcal{K})$. This means that there exists a pair of constants $a, b \in \Gamma_C$ such that $\mathsf{ga}(R, a, b) \in \mathit{chase}(\mathcal{K})$ and $A(a) \notin \mathit{chase}(\mathcal{K})$. However, such a situation would trigger the chase rule **cr3**, since $\exists R \sqsubseteq A$ would be applicable to $\mathsf{ga}(R, a, b)$ in $\mathit{chase}(\mathcal{K})$ and Proposition 6 ensures that such a PI would be applied at some step in the construction of the chase, thus causing the insertion of $A(a)$ in $\mathit{chase}(\mathcal{K})$, hence contradicting the assumption.

Furthermore, assume by contradiction that a PI of the form $\exists R_1 \sqsubseteq \exists R_2 \in \mathcal{T}_p$, where R_1 and R_2 are basic roles, is not satisfied by $\mathit{can}(\mathcal{K})$. This means that there exists a pair of constants $a, b \in \Gamma_C$ such that $\mathsf{ga}(R_1, a, b) \in \mathit{chase}(\mathcal{K})$ and there does not exist a constant $c \in \Gamma_C$ such that $\mathsf{ga}(R, a, c) \in \mathit{chase}(\mathcal{K})$. However, such a situation would trigger the chase rule **cr4** since $\exists R_1 \sqsubseteq \exists R_2$ would be applicable to $\mathsf{ga}(R_1, a, b)$ in $\mathit{chase}(\mathcal{K})$ and Proposition 6 ensures that such a PI would be applied at some step in the construction of the chase, thus causing the insertion of $\mathsf{ga}(R, a, d)$ in $\mathit{chase}(\mathcal{K})$, where $d \in \Gamma_C$ follows lexicographically all constants occurring in $\mathit{chase}(\mathcal{K})$ before the execution of **cr4**, hence contradicting the assumption.

Finally, assume by contradiction that a PI of the form $R_1 \sqsubseteq R_2 \in \mathcal{T}_p$, where R_1 and R_2 are basic roles, is not satisfied by $\mathit{can}(\mathcal{K})$ (notice that this kind of PIs needs to be taken into account only for *DL-Lite*$_\mathcal{R}$ KBs). This means that there exists a pair of constants $a, b \in \Gamma_C$ such that $\mathsf{ga}(R_1, a, b) \in \mathit{chase}(\mathcal{K})$ and $\mathsf{ga}(R_2, a, b) \notin \mathit{chase}(\mathcal{K})$. However, such a situation would trigger the chase rule **cr5**, since $R_1 \sqsubseteq R_2$ would be applicable to $\mathsf{ga}(R_1, a, b)$ in $\mathit{chase}(\mathcal{K})$ and Proposition 6 ensures that such a PI would be applied at some step in the construction of the chase, thus causing the insertion of $\mathsf{ga}(R_2, a, b)$ in $\mathit{chase}(\mathcal{K})$, hence contradicting the assumption. □

As a consequence of Lemma 7, every *DL-Lite*$_\mathcal{R}$ or *DL-Lite*$_\mathcal{F}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with only positive inclusions in the TBox, i.e., such that $\mathcal{T} = \mathcal{T}_p$, is always satisfiable, since we can always construct $\mathit{can}(\mathcal{K})$ which is a model for $\mathcal{K}$. Now, one might ask if and how $\mathit{can}(\mathcal{K})$ can

be exploited for checking the satisfiability of a KB with also negative inclusions and, for *DL-Lite*$_{\mathcal{F}}$ KBs, functionality assertions.

As for functionality assertions, the following lemma shows that, to establish that they are satisfied by $can(\mathcal{K})$, we have to simply verify that the interpretation $db(\mathcal{A})$ satisfies them (and vice-versa).

LEMMA 8. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{\mathcal{F}}$ KB, and let $\mathcal{T}_f$ be the set of functionality assertions in $\mathcal{T}$. Then, $can(\mathcal{K})$ is a model of $\langle \mathcal{T}_f, \mathcal{A} \rangle$ if and only if $db(\mathcal{A})$ is a model of $\langle \mathcal{T}_f, \mathcal{A} \rangle$.*

Proof. "$\Rightarrow$" We show that $db(\mathcal{A}) \models \langle \mathcal{T}_f, \mathcal{A} \rangle$ if $can(\mathcal{K}) \models \langle \mathcal{T}_f, \mathcal{A} \rangle$. This can be easily seen by observing that $\mathcal{A} \subseteq chase(\mathcal{K})$, and therefore if a membership assertion in $\mathcal{A}$ or a functionality assertion in $\mathcal{T}_f$ is satisfied by $can(\mathcal{K})$, it is also satisfied by $db(\mathcal{A})$ (notice in particular that $\Delta^{db(\mathcal{A})} \subseteq \Delta^{can(\mathcal{K})}$).

"$\Leftarrow$" We show that $can(\mathcal{K}) \models \langle \mathcal{T}_f, \mathcal{A} \rangle$ if $db(\mathcal{A}) \models \langle \mathcal{T}_f, \mathcal{A} \rangle$. By virtue of the correspondence between $can(\mathcal{K})$ and $chase(\mathcal{K})$, we proceed by induction on the construction of $chase(\mathcal{K})$.

Base step. We have that $chase_0(\mathcal{K}) = \mathcal{A}$, and since $db(\mathcal{A}) \models \langle \mathcal{T}_f, \mathcal{A} \rangle$, it follows that $can_0(\mathcal{K}) \models \langle \mathcal{T}_f, \mathcal{A} \rangle$.

Inductive step. Let us assume by contradiction that for some $i \geq 0$, $can_i(\mathcal{K})$ is a model of $\langle \mathcal{T}_f, \mathcal{A} \rangle$ and $can_{i+1}(\mathcal{K})$ is not. Notice that **cr2** and **cr4** are the only rules which may lead to a violation of a functionality assertion in $can_{i+1}(\mathcal{K})$. Let us consider first rule **cr2**, and assume that $chase_{i+1}(\mathcal{K})$ is obtained by applying **cr2** to $chase_i(\mathcal{K})$. This means that a PI of the form $A \sqsubseteq \exists R$, where A is an atomic concept and R is a basic role, is applied in $chase_i(\mathcal{K})$ to a membership assertion of the form $A(a)$, such that there does not exists $c \in \Gamma_C$ such that $\mathsf{ga}(R, a, c) \in chase_i(\mathcal{K})$. Therefore, $chase_{i+1}(\mathcal{K}) = chase_i(\mathcal{K}) \cup \mathsf{ga}(R, a, a_{new})$, where $a_{new} \in \Gamma_C$ follows lexicographically all constants occurring in $chase_i(\mathcal{K})$. Now, if $can_{i+1}(\mathcal{K})$ is not a model of $\langle \mathcal{T}_f, \mathcal{A} \rangle$, there must exist (at least) a functionality assertion α which is not satisfied by $can_{i+1}(\mathcal{K})$. However,

- in the case in which $\alpha = (\mathsf{funct}\ R)$, for α to be violated, there must exist two pairs of objects $(x, y), (x, z) \in R^{can_{i+1}(\mathcal{K})}$ such that $y \neq z$; however, we have that $(a, a_{new}) \in R^{can_{i+1}(\mathcal{K})}$ and $a \notin \exists R^{can_i(\mathcal{K})}$, since by applicability of $A \sqsubseteq \exists R$ in $chase_i(\mathcal{K})$ it follows that there does not exist a constant $b \in \Gamma_C$ such that $\mathsf{ga}(R, a, b) \in chase_i(\mathcal{K})$. Therefore, there exists no pair $(a, a') \in R^{can_{i+1}(\mathcal{K})}$ such that $a' \neq a_{new}$. Hence, we would conclude that the pairs $(x, y), (x, z)$ we are looking for are such that $(x, y), (x, z) \in R^{can_i(\mathcal{K})}$, but this would lead to a contradiction;
- in the case in which $\alpha = (\mathsf{funct}\ R^-)$, for α to be violated, there must exist two pairs of objects $(y, x), (z, x) \in R^{can_{i+1}(\mathcal{K})}$ such that

$y \neq z$; since a_{new} is a fresh constant, not occurring in $chase_i(\mathcal{K})$, we can conclude that there exists no pair (a', a_{new}), with $a' \neq a$, such that $\mathsf{ga}(R, a', a_{new}) \in chase_i(\mathcal{K})$, and therefore, there exists no pair $(a', a_{new}) \in R^{can_{i+1}(\mathcal{K})}$. Hence, we would conclude that the pairs $(y, x), (z, x)$ we are looking for, are such that $(y, x), (z, x) \in R^{can_i(\mathcal{K})}$, but this would lead to a contradiction;

– in the case in which $\alpha = (\mathsf{funct}\ R')$, with $R' \neq R$, we would conclude that α is not satisfied already in $can_i(\mathcal{K})$, but this would lead to a contradiction.

With an almost identical argument we can prove the inductive step also in the case in which $chase_{i+1}(\mathcal{K})$ is obtained by applying **cr4** to $chase_i(\mathcal{K})$. □

3.1.2. *NI-closure*

Let us now consider negative inclusions. In particular, we look for a property which is analogous to Lemma 8 for the case of NIs. Notice that, in this case, even if $db(\mathcal{A})$ satisfies the NIs asserted in the KB $\mathcal{K} = \langle \mathcal{T}, A \rangle$, we have that $can(\mathcal{K})$ may not satisfy $\mathcal{K}$. For example, if $\mathcal{T}$ contains the inclusion assertions $A_1 \sqsubseteq A_2$ and $A_2 \sqsubseteq \neg A_3$, and $\mathcal{A}$ contains the membership assertions $A_1(a)$ and $A_3(a)$, it is easy to see that $db(\mathcal{A}) \models A_2 \sqsubseteq \neg A_3$, but $can(\mathcal{K}) \not\models A_2 \sqsubseteq \neg A_3$.

However, as suggested by the simple example above, we get that to find the property we are looking for, we need to properly take into account the interaction between positive and negative inclusions. To this aim we construct a special TBox by closing the NIs with respect to the PIs.

DEFINITION 9. Let $\mathcal{T}$ be a *DL-Lite*$_\mathcal{R}$ or a *DL-Lite*$_\mathcal{F}$ TBox. We call *NI-closure of* $\mathcal{T}$, denoted by $cln(\mathcal{T})$, the TBox defined inductively as follows:

1. all negative inclusion assertions in $\mathcal{T}$ are also in $cln(\mathcal{T})$;
2. all functionality assertions in $\mathcal{T}$ are also in $cln(\mathcal{T})$;
3. if $B_1 \sqsubseteq B_2$ is in $\mathcal{T}$ and $B_2 \sqsubseteq \neg B_3$ or $B_3 \sqsubseteq \neg B_2$ is in $cln(\mathcal{T})$, then also $B_1 \sqsubseteq \neg B_3$ is in $cln(\mathcal{T})$;
4. if $R_1 \sqsubseteq R_2$ is in $\mathcal{T}$ and $\exists R_2 \sqsubseteq \neg B$ or $B \sqsubseteq \neg \exists R_2$ is in $cln(\mathcal{T})$, then also $\exists R_1 \sqsubseteq \neg B$ is in $cln(\mathcal{T})$;
5. if $R_1 \sqsubseteq R_2$ is in $\mathcal{T}$ and $\exists R_2^- \sqsubseteq \neg B$ or $B \sqsubseteq \neg \exists R_2^-$ is in $cln(\mathcal{T})$, then also $\exists R_1^- \sqsubseteq \neg B$ is in $cln(\mathcal{T})$;

6. if $R_1 \sqsubseteq R_2$ is in $\mathcal{T}$ and $R_2 \sqsubseteq \neg R_3$ or $R_3 \sqsubseteq \neg R_2$ is in $cln(\mathcal{T})$, then also $R_1 \sqsubseteq \neg R_3$ is in $cln(\mathcal{T})$.

7. (a) in the case in which $\mathcal{T}$ is a *DL-Lite*$_\mathcal{F}$ TBox, if one of the assertions $\exists R \sqsubseteq \neg\exists R$, or $\exists R^- \sqsubseteq \neg\exists R^-$ is in $cln(\mathcal{T})$, then both such assertions are in $cln(\mathcal{T})$;
   (b) in the case in which $\mathcal{T}$ s a *DL-Lite*$_\mathcal{R}$ TBox, if one of the assertions $\exists R \sqsubseteq \neg\exists R$, $\exists R^- \sqsubseteq \neg\exists R^-$, or $R \sqsubseteq \neg R$ is in $cln(\mathcal{T})$, then all three such assertions are in $cln(\mathcal{T})$.

Notice that in the construction of the NI-closure of *DL-Lite*$_\mathcal{F}$ TBoxes, we make use only of Rules 1, 2, 3, and 7(a) whereas for *DL-Lite*$_\mathcal{R}$ TBoxes, we make use only of Rules 1, 3, 4, 5, 6, and 7(b).

The following lemma shows that $cln(\mathcal{T})$ for *DL-Lite*$_\mathcal{R}$ KBs does not imply new negative inclusions not implied by $\mathcal{T}$.

LEMMA 10. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox, and α a negative inclusion assertion. We have that, if $cln(\mathcal{T}) \models \alpha$, then $\mathcal{T} \models \alpha$.*

Proof. To prove the claim it is sufficient to observe that all assertions contained in $cln(\mathcal{T})$ are logically implied by $\mathcal{T}$. □

An analogous property holds for *DL-Lite*$_\mathcal{F}$ KBs, where also implication of functionality assertions is taken into account.

LEMMA 11. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{F}$ TBox, and α a negative inclusion assertion or a functionality assertion. We have that, if $cln(\mathcal{T}) \models \alpha$, then $\mathcal{T} \models \alpha$.*

Proof. Analogous to the proof of Lemma 10. □

We are now ready to show that, provided we have computed $cln(\mathcal{T})$, the analogous of Lemma 7 and Lemma 8 holds also for NIs.

LEMMA 12. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ KB. Then, $can(\mathcal{K})$ is a model of $\mathcal{K}$ if and only if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$.*

Proof. We provide the proof for the case of *DL-Lite*$_\mathcal{R}$ KBs. The case of *DL-Lite*$_\mathcal{F}$ KBs can be proved in an analogous way.

"⇒" By construction, $db(\mathcal{A})$ cannot contradict a membership assertion in $\mathcal{A}$. Moreover, since $can(\mathcal{K})$ is a model of $\mathcal{K}$ and, by Lemma 10, each assertion in $cln(\mathcal{T})$ is logically implied by $\mathcal{K}$, then $can(\mathcal{K})$ is a model of $cln(\mathcal{T})$. Notice that $A^{db(\mathcal{A})} = A^{can_0(\mathcal{K})} \subseteq A^{can(\mathcal{K})}$ for every atomic concept A in $\mathcal{K}$, and similarly $P^{db(\mathcal{A})} = P^{can_0(\mathcal{K})} \subseteq P^{can(\mathcal{K})}$ for every atomic role P in $\mathcal{K}$. Now, considering that the structure of

NIs (and also functionalities for *DL-Lite*$_{\mathcal{F}}$) is such that they cannot be contradicted by restricting the extension of atomic concepts and roles, we can conclude that $db(\mathcal{A})$ is a model of $cln(\mathcal{T})$.

"$\Leftarrow$" We now prove that if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A}\rangle$, then $can(\mathcal{K})$ is a model of $\mathcal{K}$. From Lemma 7 it follows that $can(\mathcal{K})$ is a model of $\langle \mathcal{T}_p, \mathcal{A}\rangle$, where $\mathcal{T}_p$ is the set of PIs in $\mathcal{T}$[7]. Hence, it remains to prove that $can(\mathcal{K})$ is a model of $\langle \mathcal{T} \setminus \mathcal{T}_p, \mathcal{A}\rangle$. We show this by proving that $can(\mathcal{K})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A}\rangle$ (notice that $\mathcal{T} \setminus \mathcal{T}_p$ is contained in $cln(\mathcal{T})$). The proof is by induction on the construction of $chase(\mathcal{K})$.

Base step. By construction, $chase_0(\mathcal{K}) = \mathcal{A}$, and therefore $A^{can_0(\mathcal{K})} = A^{db(\mathcal{A})}$ for every atomic concept A in $\mathcal{K}$, and $P^{can_0(\mathcal{K})} = P^{db(\mathcal{A})}$ for every atomic role P in $\mathcal{K}$. Hence, by the assumption that $db(\mathcal{A}) \models \langle cln(\mathcal{T}), \mathcal{A}\rangle$, it follows that $can_0(\mathcal{K})$ is a model for $\langle cln(\mathcal{T}), \mathcal{A}\rangle$.

Inductive step. Let us assume by contradiction that $can_i(\mathcal{K})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A}\rangle$ and $can_{i+1}(\mathcal{K})$ is not, and that $chase_{i+1}(\mathcal{K})$ is obtained from $chase_i(\mathcal{K})$ by execution of the rule **cr1**. According to **cr1**, a PI of the form $A_1 \sqsubseteq A_2$, where A_1 and A_2 are atomic concepts in $\mathcal{T}$, is applied in $chase_i(\mathcal{K})$ to a membership assertion of the form $A_1(d)$, such that $A_2(d) \notin chase_i(\mathcal{K})$. Therefore $chase_{i+1}(\mathcal{K}) = chase_i(\mathcal{K}) \cup \{A_2(d)\}$ (notice that this means that $d \in A_2^{can_{i+1}(\mathcal{K})}$). Now, if $can_{i+1}(\mathcal{K})$ is not a model of $cln(\mathcal{T})$, there must exist a NI in $cln(\mathcal{T})$ of the form $A_2 \sqsubseteq \neg A_3$, where A_3 is an atomic concept, (or $A_2 \sqsubseteq \neg\exists R$, where R is a basic role) or of the form $A_3 \sqsubseteq \neg A_2$ (resp., $\exists R \sqsubseteq \neg A_2$), such that $A_3(d) \in chase_i(\mathcal{K})$ (resp., there exists a constant c such that $\mathsf{ga}(R, a, c) \in chase_i(\mathcal{K})$). Notice that this means that $d \in A_3^{can_i(\mathcal{K})}$ (resp., $d \in \exists R^{can_i(\mathcal{K})}$). It is easy to see that, if such a NI exists, then also $A_1 \sqsubseteq \neg A_3$ (resp., $A_1 \sqsubseteq \neg\exists R$) belongs to $cln(\mathcal{T})$, according to NI-closure rule 3. Since $chase_{i+1}(\mathcal{K}) = chase_i(\mathcal{K}) \cup \{A_2(d)\}$, then $A_1 \sqsubseteq \neg A_3$ (resp., $A_1 \sqsubseteq \neg\exists R$) is not satisfied already by $can_i(\mathcal{K})$, if $A_3 \neq A_2$. If $A_3 = A_2$, we need to again consider NI-closure rule 3, according to which, from the fact that $A_1 \sqsubseteq A_2$ in $\mathcal{T}_p$, and $A_1 \sqsubseteq \neg A_2$ in $cln(\mathcal{T})$, it follows that $A_1 \sqsubseteq \neg A_1$ is in $cln(\mathcal{T})$, which is therefore not satisfied already by $can_i(\mathcal{K})$. In both cases, we have thus contradicted the assumption that $can_i(\mathcal{K})$ is a model of $cln(\mathcal{T})$. With an almost identical argument we can prove the inductive step also in those cases in which $chase_{i+1}(\mathcal{K})$ is obtained from $chase_i(\mathcal{K})$ by executing rule **cr3** or rule **cr5** (in this last case, in particular, we need to use in the proof NI-closure rules 4, 5 and 6). As for the cases in which $chase_{i+1}(\mathcal{K})$ is obtained from $chase_i(\mathcal{K})$ by applying rule **cr2**, we proceed as follows (for rule **cr4** the proof is analogous). According to **cr2**, a PI of the

[7] Notice that, for the case of *DL-Lite*$_{\mathcal{F}}$ KBs, from Lemma 8 it follows that $can(\mathcal{K})$ is a model of $\langle \mathcal{T}_f, \mathcal{A}\rangle$, where $\mathcal{T}_f$ is the set of functionality assertions in $\mathcal{K}$.

form $A \sqsubseteq \exists R$, where A is an atomic concept in $\mathcal{T}$, and R is a basic role in $\mathcal{T}$, is applied in $chase_i(\mathcal{K})$ to a membership assertion $A(d)$ such that there does not exist $f \in \Gamma_C$ such that $\mathsf{ga}(R, d, f) \in chase_i(\mathcal{K})$. Therefore $chase_{i+1}(\mathcal{K}) = chase_i(\mathcal{K}) \cup \{\mathsf{ga}(R, d, e)\}$, where e follows lexicographically all constants appearing in $chase_i(\mathcal{K})$ (notice that this means that $d \in \exists R^{can_{i+1}(\mathcal{K})}$). Now, if $can_{i+1}(\mathcal{K})$ is not a model of $cln(\mathcal{T})$, there must exist a NI in $cln(\mathcal{T})$ of the form $\exists R \sqsubseteq \neg B$, where B is a basic concept, or of the form $\exists R^- \sqsubseteq \neg \exists R^-$, or of the form $R \sqsubseteq \neg R$. As for the first form of NI, we can reach a contradiction as done above for the case of execution of chase rule **cr1**. As for the last two forms of NIs, according to NI-closure rule 7(b), we have that if (at least) one of these NIs is in $cln(\mathcal{T})$, then also $\exists R \sqsubseteq \neg \exists R$ is in $cln(\mathcal{T})$, and thus we can again reason on a NI of the first form to reach a contradiction. □

The following corollary is an interesting consequence of the lemma above.

COROLLARY 13. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ TBox, and α a negative inclusion assertion or a functionality assertion. We have that, if $\mathcal{T} \models \alpha$, then $cln(\mathcal{T}) \models \alpha$.*

Proof. We first consider the case in which α is a NI. We prove the claim by contradiction. Let us assume that $\mathcal{T} \models \alpha$ and $cln(\mathcal{T}) \not\models \alpha$. We show that from $cln(\mathcal{T}) \not\models \alpha$ one can construct a model of $\mathcal{T}$ which does not satisfy α, thus obtaining a contradiction.

Let us assume that $\alpha = A_1 \sqsubseteq \neg A_2$, where A_1 and A_2 are atomic concepts in $\mathcal{T}$, and consider the *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A} = \{A_1(a), A_2(a)\}$. We show that $can(\mathcal{K})$ is the model we are looking for, i.e., $can(\mathcal{K}) \models \mathcal{T}$ but $can(\mathcal{K}) \not\models \alpha$. The last property follows trivially by the form of $\mathcal{A}$. Hence, in the following we concentrate on proving that $can(\mathcal{K}) \models \mathcal{T}$.

We recall that $db(\mathcal{A})$ is such that $A_1^{db(\mathcal{A})} = \{a\}$, $A_2^{db(\mathcal{A})} = \{a\}$, $A^{db(\mathcal{A})} = \emptyset$ for each atomic concept $A \in \mathcal{T}$ such that $A \neq A_1$ and $A \neq A_2$, and $R^{db(\mathcal{A})} = \emptyset$ for each atomic role $R \in \mathcal{T}$. Therefore, the only NIs that can be violated by $db(\mathcal{A})$ are $A_1 \sqsubseteq \neg A_2$, $A_2 \sqsubseteq \neg A_1$, $A_1 \sqsubseteq \neg A_1$, and $A_2 \sqsubseteq \neg A_2$. By assumption, we have that $cln(\mathcal{T}) \not\models A_1 \sqsubseteq \neg A_2$, and therefore also $cln(\mathcal{T}) \not\models A_2 \sqsubseteq \neg A_1$. From this, it follows also that $cln(\mathcal{T}) \not\models A_1 \sqsubseteq \neg A_1$ and $cln(\mathcal{T}) \not\models A_2 \sqsubseteq \neg A_2$, since either $A_1 \sqsubseteq \neg A_1$ or $A_2 \sqsubseteq \neg A_2$ logically implies $A_1 \sqsubseteq \neg A_2$. Therefore, we can conclude that $db(\mathcal{A}) \models cln(\mathcal{T})$ and hence $db(\mathcal{A}) \models \langle cln(\mathcal{T}), \mathcal{A} \rangle$, for either the case in which $\mathcal{T}$ is a *DL-Lite*$_\mathcal{R}$ KB (and hence $cln(\mathcal{T})$ implies only NIs), or the case in which $\mathcal{T}$ is a *DL-Lite*$_\mathcal{F}$ KB, since it is obvious that, being $\mathcal{A} = \{A_1(a), A_2(a)\}$, $db(\mathcal{A})$ cannot violate functionality

assertions. Then, from Lemma 12 it follows that $can(\mathcal{K})$ is a model of $\mathcal{K}$.

Proceeding analogously as done above, we can easily prove the claim in those cases in which $\alpha = A \sqsubseteq \neg\exists R$, $\alpha = \exists R \sqsubseteq \neg A$, $\alpha = \exists R_1 \sqsubseteq \neg\exists R_2$, or $\alpha = R_1 \sqsubseteq \neg R_2$ (the last one only for *DL-Lite*$_\mathcal{R}$ TBoxes).

The proof for the case in which α is a functionality assertion of the form $(\mathsf{funct}\ R)$ can be obtained in an analogous way, by constructing the canonical interpretation starting from an ABox with the assertions $\mathsf{ga}(R, a, b)$ and $\mathsf{ga}(R, a, c)$. □

3.1.3. *FOL-reducibility*

Before providing the main theorems of this subsection, we need also the following property, which asserts that to establish satisfiability of a knowledge base, we can resort to constructing the canonical interpretation.

LEMMA 14. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ KB. Then, $can(\mathcal{K})$ is a model of $\mathcal{K}$ if and only if $\mathcal{K}$ is satisfiable.*

Proof. We prove the lemma for *DL-Lite*$_\mathcal{R}$ KBs. The following proof can be easily adapted to prove the lemma also for *DL-Lite*$_\mathcal{F}$ KBs.

"$\Rightarrow$" If $can(\mathcal{K})$ is a model for $\mathcal{K}$, then $\mathcal{K}$ is obviously satisfiable.

"$\Leftarrow$" We prove this direction by showing that if $can(\mathcal{K})$ is not a model of $\mathcal{K}$, then $\mathcal{K}$ is unsatisfiable. By Lemma 12 ("if" direction), it follows that $db(\mathcal{A})$ is not a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$, and therefore $db(\mathcal{A}) \not\models cln(\mathcal{T})$. This means that there exists a NI α such that $db(\mathcal{A}) \not\models \alpha$ and $cln(\mathcal{T}) \models \alpha$, and hence by Lemma 10 $\mathcal{T} \not\models \alpha$. Let us assume that α is of the form $B_1 \sqsubseteq \neg B_2$, where B_1 and B_2 are basic concepts (resp., α is of the form $R_1 \sqsubseteq \neg R_2$, where R_1 and R_2 are atomic roles). Then, there exists $a \in \Delta^{db(\mathcal{A})}$ such that $a \in B_1^{db(\mathcal{A})}$ and $a \in B_2^{db(\mathcal{A})}$ (resp., there exist $a, b \in \Delta^{db(\mathcal{A})}$ such that $(a, b) \in R_1^{db(\mathcal{A})}$ and $(a, b) \in R_2^{db(\mathcal{A})}$). Let us now assume by contradiction that a model $\mathcal{M} = \langle \Delta^\mathcal{M}, \cdot^\mathcal{M} \rangle$ of $\mathcal{K}$ exists. For each model $\mathcal{M}$, we can construct a homomorphism ψ from $\Delta^{db(\mathcal{A})}$ to $\Delta^\mathcal{M}$ such that $\psi(c) = c^\mathcal{M}$ for each constant c occurring in $\mathcal{A}$ (notice that $\mathcal{M}$ assigns a distinct object to each such constant c, since $\mathcal{M} \models \mathcal{A}$). From the fact that $\mathcal{M}$ satisfies membership assertions in $\mathcal{A}$, it easily follows that $\psi(a) \in B_1^\mathcal{M}$ and $\psi(a) \in B_2^\mathcal{M}$ (resp., $(\psi(a), \psi(b)) \in R_1^\mathcal{M}$ and $(\psi(a), \psi(b)) \in R_2^\mathcal{M}$), but this makes the NI $B_1 \sqsubseteq \neg B_2$ (resp., $R_1 \sqsubseteq \neg R_2$) be violated also in $\mathcal{M}$, thus contradicting the fact that $\mathcal{M}$ is a model, since a model cannot violate a NI that is logically implied by $\mathcal{T}$. □

Notice that, the construction of $can(\mathcal{K})$ is in general neither convenient nor possible, since $can(\mathcal{K})$ may be infinite. However, by simply

combining Lemma 12 and Lemma 14, we obtain the notable result that to check satisfiability of a knowledge base, it is sufficient (and necessary) to look at $db(\mathcal{A})$ (provided we have computed $cln(\mathcal{T})$). More precisely, the next theorem shows that a contradiction on a *DL-Lite*$_\mathcal{R}$ or a *DL-Lite*$_\mathcal{F}$ KB may hold only if a membership assertion in the ABox contradicts a functionality assertion or a NI implied by the closure $cln(\mathcal{T})$.

THEOREM 15. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ KB. Then, $\mathcal{K}$ is satisfiable if and only if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$.*

Proof. "$\Rightarrow$" If $\mathcal{K}$ is satisfiable, from Lemma 14 ("only-if" direction), it follows that $can(\mathcal{K})$ is a model of $\mathcal{K}$, and therefore, from Lemma 12 ("only-if" direction), it follows that $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$.

"$\Leftarrow$" If $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$, from Lemma 12 ("if" direction), it follows that $can(\mathcal{K})$ is a model of $\mathcal{K}$, and therefore $\mathcal{K}$ is satisfiable. □

At this point, it is not difficult to show that verifying if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$ can be done by simply evaluating a suitable boolean FOL query over $db(\mathcal{A})$ itself. In particular we define a translation function δ from assertions in $cln(\mathcal{T})$ to FOL formulas, as follows:

$$\begin{array}{lcl}
\delta((\mathsf{funct}\ P)) & = & \exists x, y, z.P(x, y) \wedge P(x, z) \wedge y \neq z \\
\delta((\mathsf{funct}\ P^-)) & = & \exists x, y, z.P(x, y) \wedge P(z, y) \wedge x \neq z \\
\delta(B_1 \sqsubseteq \neg B_2) & = & \exists x.\gamma_1(x) \wedge \gamma_2(x) \\
\delta(R_1 \sqsubseteq \neg R_2) & = & \exists x, y.\rho_1(x, y) \wedge \rho_2(x, y)
\end{array}$$

where in the last equations $\gamma_i(x) = A_i(x)$ if $B_i = A_i$, $\gamma_i(x) = \exists y_i.P_i(x, y_i)$ if $B_i = \exists P_i$, and $\gamma_i(x) = \exists y_i.P_i(y_i, x)$ if $B_i = \exists P_i^-$; and $\rho_i(x, y) = P_i(x, y)$ if $R_i = P_i$, and $\rho_i(x, y) = P_i(y, x)$ if $R_i = P_i^-$.

The algorithm Consistent, described in Figure 1, takes as input a *DL-Lite*$_\mathcal{R}$ or a *DL-Lite*$_\mathcal{F}$ KB, computes $db(\mathcal{A})$ and $cln(\mathcal{T})$, and evaluates over $db(\mathcal{A})$ the boolean FOL query obtained by taking the union of all FOL formulas returned by the application of the above function δ to every assertion in $cln(\mathcal{T})$. In the algorithm, the symbol $\bot$ indicates a predicate whose evaluation is *false* in every interpretation. Therefore, in the case in which neither functionality assertions nor negative inclusion assertions occur in $\mathcal{K}$, $q_{unsat}^{db(\mathcal{A})} = \bot^{db(\mathcal{A})}$, and therefore Consistent($\mathcal{K}$) returns *true*.

LEMMA 16. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ KB. Then, the algorithm Consistent($\mathcal{K}$) terminates, and $\mathcal{K}$ is satisfiable if and only if Consistent($\mathcal{K}$) = true.*

Algorithm Consistent($\mathcal{K}$)
Input: *DL-Lite* knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: *true* if $\mathcal{K}$ is satisfiable, *false* otherwise
begin
  $q_{unsat} = \bot$;
  for each $\alpha \in cln(\mathcal{T})$ **do**
    $q_{unsat} = q_{unsat} \vee \delta(\alpha)$;
  if $q_{unsat}^{db(\mathcal{A})} = \emptyset$ **return** *true*;
  else return *false*;
end

Figure 1. The algorithm Consistent

Proof. Since $cln(\mathcal{T})$ is a finite set of membership and functionality assertions, the algorithm terminates. By Theorem 15, we have that $db(\mathcal{A})$ is a model of all assertions in $cln(\mathcal{T})$ if and only if $\mathcal{K}$ is satisfiable. The query q_{unsat} verifies whether there exists an assertion α that is violated in $db(\mathcal{A})$, by expressing its negation as a FOL formula $\delta(\alpha)$ and evaluating it in $db(\mathcal{A})$. □

As a direct consequence of Lemma 16, we get:

THEOREM 17. *Knowledge base satisfiability in DL-Lite$_\mathcal{R}$ and DL-Lite$_\mathcal{F}$ is FOL-reducible.*

EXAMPLE 18. We now check satisfiability of the *DL-Lite$_{core}$* KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in Example 1. To this aim, we first compute $cln(\mathcal{T})$, which is as follows:

$$\begin{aligned} Professor &\sqsubseteq \neg Student \\ \exists TeachesTo^- &\sqsubseteq \neg Professor \\ \exists HasTutor^- &\sqsubseteq \neg Student. \end{aligned}$$

Next, we apply the translation function δ to each NI above, getting:

$$\begin{aligned} &\delta(Professor \sqsubseteq \neg Student) = \exists x.Professor(x) \wedge Student(x) \\ &\delta(\exists TeachesTo^- \sqsubseteq \neg Professor) = \exists x.(\exists y.TeachesTo(y,x)) \wedge Professor(x) \\ &\delta(\exists HasTutor^- \sqsubseteq \neg Student) = \exists x.(\exists y.HasTutor(y,x)) \wedge Student(x). \end{aligned}$$

The union of such queries is q_{unsat}, which evaluated over $db(\mathcal{A})$ returns false, thus showing satisfiability of $\mathcal{K}$.

As a further example, consider now the *DL-Lite$_\mathcal{R}$* TBox $\mathcal{T}'$ obtained from $\mathcal{T}$ by adding the inclusion assertion $HasTutor^- \sqsubseteq TeachesTo$. In this case $cln(\mathcal{T}')$ includes $cln(\mathcal{T})$ plus the following NIs:

$$\begin{aligned} \exists HasTutor &\sqsubseteq \neg Professor \\ \exists TeachesTo &\sqsubseteq \neg Student. \end{aligned}$$

So q'_{unsat} includes the disjuncts of q_{unsat} plus the following:

$$\delta(\exists HasTutor \sqsubseteq \neg Professor) = \exists x.(\exists y.HasTutor(x,y)) \wedge Professor(x)$$
$$\delta(\exists TeachesTo \sqsubseteq \neg Student) = \exists x.(\exists y.TeachesTo(x,y)) \wedge Student(x).$$

Since $(q'_{unsat})^{db(\mathcal{A})}$ is false, we conclude that $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A} \rangle$ is satisfiable.

Finally, if we instead add the functionality assertion (funct *HasTutor*) to $\mathcal{T}$, we obtain a *DL-Lite*$_{\mathcal{F}}$ TBox $\mathcal{T}''$, whose NI-closure $cln(\mathcal{T}'')$ includes $cln(\mathcal{T})$ plus (funct *HasTutor*).

In this case, q''_{unsat} includes the disjuncts of q_{unsat} plus:

$$\delta((\text{funct } HasTutor)) = \exists x, y, z.HasTutor(x,y) \wedge HasTutor(x,z) \wedge y \neq z.$$

Again, $(q''_{unsat})^{db(\mathcal{A})}$ is false, and hence also $\mathcal{K}'' = \langle \mathcal{T}'', \mathcal{A} \rangle$ is satisfiable.
■

3.2. Logical implication

We start by showing that both instance checking and subsumption can be reduced to knowledge base satisfiability. We first consider the problem of instance checking for concept expressions, and provide a suitable reduction from this problem to knowledge base satisfiability.

THEOREM 19. *Let $\mathcal{K}$ be either a DL-Lite$_{\mathcal{R}}$ or a DL-Lite$_{\mathcal{F}}$ KB, C a general concept, d a constant appearing in $\mathcal{K}$, and $\hat{A}$ an atomic concept not appearing in $\mathcal{K}$. Then $\mathcal{K} \models C(d)$ if and only if the KB*

$$\mathcal{K}_{C(d)} = \langle \mathcal{K} \cup \{\hat{A} \sqsubseteq \neg C\},\ \{\hat{A}(d)\} \rangle$$

is unsatisfiable.

Proof. "⇒" Suppose that $\mathcal{K} \models C(d)$, but there exists a model $\mathcal{M}'$ of $\mathcal{K}_{C(d)}$. Then $\mathcal{M}' \models \hat{A}(d)$ and $\mathcal{M}' \models \hat{A} \sqsubseteq \neg C$. But then $\mathcal{M}' \models \neg C(d)$. Observe that $\mathcal{M}'$ is a model of $\mathcal{K}$, hence we get a contradiction.

"⇐" Suppose that $\mathcal{K}_{C(d)}$ is unsatisfiable, but there exists a model $\mathcal{M}$ of $\mathcal{K}$ such that $\mathcal{M} \models \neg C(d)$. Then we can define an interpretation $\mathcal{M}'$ of $\mathcal{K}_{C(d)}$ that interprets all constants, concept and roles in $\mathcal{K}$ as before, and assigns to $\hat{A}$ (which does not appear in $\mathcal{K}$) the extension $\hat{A}^{\mathcal{M}'} = \{d^{\mathcal{M}}\}$. Now, $\mathcal{M}'$ is still a model of $\mathcal{K}$, and moreover we have that $\mathcal{M}' \models \hat{A}(d)$ and $\mathcal{M}' \models \hat{A} \sqsubseteq \neg C$, hence $\mathcal{M}'$ is a model of $\mathcal{K}_{C(d)}$. Thus we get a contradiction. □

The analogous of the above theorem holds for the problem of instance checking for role expressions. We first consider *DL-Lite*$_{\mathcal{R}}$ KBs.

THEOREM 20. *Let $\mathcal{K}$ be a DL-Lite$_\mathcal{R}$ KB, E a general role, a and b two constants appearing in $\mathcal{K}$, and $\hat{P}$ an atomic role not appearing in $\mathcal{K}$. Then $\mathcal{K} \models E(a,b)$ if and only if the KB*

$$\mathcal{K}_{E(a,b)} = \langle \mathcal{K} \cup \{\hat{P} \sqsubseteq \neg E\},\ \{\hat{P}(a,b)\}\rangle$$

is unsatisfiable.

Proof. Similar to the proof of Theorem 19. □

Also, for *DL-Lite*$_\mathcal{F}$ KBs we provide the following theorem.

THEOREM 21. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ be a DL-Lite$_\mathcal{F}$ KB, R a basic role, and a and b two constants appearing in $\mathcal{K}$. Then*

- *$\mathcal{K} \models R(a,b)$ if and only if either $\mathcal{K}$ is unsatisfiable, or $\mathsf{ga}(R,a,b) \in \mathcal{A}$.*
- *$\mathcal{K} \models \neg R(a,b)$ if and only if $\langle \mathcal{T}, \mathcal{A} \cup \{\mathsf{ga}(R,a,b)\}\rangle$ is unsatisfiable.*

Proof. The second item is obvious. As for the first item, the "if" direction is trivial, while for the "only-if" direction, assume that $\mathcal{K}$ is satisfiable and $\mathsf{ga}(R,a,b) \notin \mathcal{A}$. From the fact that $\mathcal{K}$ is satisfiable, it follows (see Lemma 14) that $can(\mathcal{K}) \models \mathcal{K}$. From the construction of $can(\mathcal{K})$ for a *DL-Lite*$_\mathcal{F}$ KB it easily follows that $can(\mathcal{K}) \models R(a,b)$ if and only if $\mathsf{ga}(R,a,b) \in \mathcal{A}$, and therefore $can(\mathcal{K}) \not\models R(a,b)$. Now, the fact that $can(\mathcal{K}) \models \mathcal{K}$ and $can(\mathcal{K}) \not\models R(a,b)$ contradicts the assumption that $\mathcal{K} \models R(a,b)$. □

We now address the subsumption problem and provide different reductions of this problem to the problem of knowledge base satisfiability. The case of subsumption between concepts is dealt with by the following theorem, and the case of subsumption between roles, is considered in the two subsequent theorems.

THEOREM 22. *Let $\mathcal{T}$ be either a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ TBox, C_1 and C_2 two general concepts, $\hat{A}$ an atomic concept not appearing in $\mathcal{T}$, and d a constant. Then, $\mathcal{T} \models C_1 \sqsubseteq C_2$ if and only if the KB*

$$\mathcal{K}_{C_1 \sqsubseteq C_2} = \langle \mathcal{T} \cup \{\hat{A} \sqsubseteq C_1, \hat{A} \sqsubseteq \neg C_2\},\ \{\hat{A}(d)\}\rangle,$$

is unsatisfiable.

Proof. "⇒" Suppose that $\mathcal{T} \models C_1 \sqsubseteq C_2$, but there exists a model $\mathcal{M}'$ of $\mathcal{K}_{C_1 \sqsubseteq C_2}$. Then $\mathcal{M}' \models \hat{A}(d)$, $\mathcal{M}' \models \hat{A} \sqsubseteq C_1$, and $\mathcal{M}' \models \hat{A} \sqsubseteq \neg C_2$. But then $\mathcal{M}' \models C_1(d)$ and $\mathcal{M}' \models \neg C_2(d)$. Observe that $\mathcal{M}'$ is a model of $\mathcal{T}$, hence we get a contradiction.

"⇐" Suppose that $\mathcal{K}_{C_1 \sqsubseteq C_2}$ is unsatisfiable, but there exists a model $\mathcal{M}$ of $\mathcal{T}$ such that $o \in C_1^{\mathcal{M}}$ and $o \notin C_2^{\mathcal{M}}$ for some object o in the domain of $\mathcal{M}$. Then we can define an interpretation $\mathcal{M}'$ of $\mathcal{K}_{C_1 \sqsubseteq C_2}$ that interprets all concepts and roles in $\mathcal{T}$ as before, and assigns to d and $\hat{A}$ (which does not appear in $\mathcal{T}$) the extensions $d^{\mathcal{M}'} = o$, $\hat{A}^{\mathcal{M}'} = \{o\}$. Now, $\mathcal{M}'$ is still a model of $\mathcal{T}$, and moreover we have that $\mathcal{M}' \models \hat{A}(d)$, $\mathcal{M}' \models \hat{A} \sqsubseteq C_1$, and $\mathcal{M}' \models \hat{A} \sqsubseteq \neg C_2$. Hence $\mathcal{M}'$ is a model of $\mathcal{K}_{C_1 \sqsubseteq C_2}$, and we get a contradiction. □

THEOREM 23. *Let $\mathcal{T}$ be a DL-Lite$_{\mathcal{R}}$ TBox, E_1 and E_2 two general roles, $\hat{P}$ an atomic role not appearing in $\mathcal{T}$, and a, b two constants. Then, $\mathcal{T} \models E_1 \sqsubseteq E_2$ if and only if the KB*

$$\mathcal{K}_{E_1 \sqsubseteq E_2} = \langle \mathcal{T} \cup \{\hat{P} \sqsubseteq E_1, \hat{P} \sqsubseteq \neg E_2\},\ \{\hat{P}(a,b)\} \rangle$$

is unsatisfiable.

Proof. "⇒" Suppose that $\mathcal{T} \models E_1 \sqsubseteq E_2$, but there exists a model $\mathcal{M}'$ of $\mathcal{K}_{E_1 \sqsubseteq E_2}$. Then $\mathcal{M}' \models \hat{P}(a,b)$, $\mathcal{M}' \models \hat{P} \sqsubseteq E_1$, and $\mathcal{M}' \models \hat{P} \sqsubseteq \neg E_2$. But then $\mathcal{M}' \models E_1(a,b)$ and $\mathcal{M}' \models \neg E_2(a,b)$. Observe that $\mathcal{M}'$ is a model of $\mathcal{T}$, hence we get a contradiction.

"⇐" Suppose that $\mathcal{K}_{E_1 \sqsubseteq E_2}$ is unsatisfiable, but there exists a model $\mathcal{M}$ of $\mathcal{T}$ such that $(o_a, o_b) \in E_1^{\mathcal{M}}$ and $(o_a, o_b) \notin E_2^{\mathcal{M}}$ for some pair of objects in the domain of $\mathcal{M}$. We first show that we can assume w.l.o.g. that o_a and o_b are distinct objects. Indeed, if $o_a = o_b$, we can construct a new model $\mathcal{M}_d$ of $\mathcal{T}$ as follows: $\Delta^{\mathcal{M}_d} = \Delta^{\mathcal{M}} \times \{1,2\}$, $A^{\mathcal{M}_d} = A^{\mathcal{M}} \times \{1,2\}$ for each atomic concept A, and $P^{\mathcal{M}_d} = (\{((o,1),(o',1)),((o,2),(o',2)) \mid (o,o') \in P^{\mathcal{M}}\} \cup U) \setminus V$, where

$$U = \begin{cases} \emptyset, & \text{if } (o_a,o_a) \notin P^{\mathcal{M}} \\ \{((o_a,1),(o_a,2)),((o_a,2),(o_a,1))\}, & \text{if } (o_a,o_a) \in P^{\mathcal{M}} \end{cases}$$
$$V = \begin{cases} \emptyset, & \text{if } (o_a,o_a) \notin P^{\mathcal{M}} \\ \{((o_a,1),(o_a,1)),((o_a,2),(o_a,2))\}, & \text{if } (o_a,o_a) \in P^{\mathcal{M}} \end{cases}$$

for each atomic role P. It is immediate to see that $\mathcal{M}_d$ is still a model of $\mathcal{T}$ containing a pair of distinct objects in $E_1^{\mathcal{M}_d}$ and not in $E_2^{\mathcal{M}_d}$.

Now, given that we can assume that $o_a \neq o_b$, we can define an interpretation $\mathcal{M}'$ of $\mathcal{K}_{E_1 \sqsubseteq E_2}$ that interprets all concepts and roles in $\mathcal{T}$ as before, and assigns to a, b, and $\hat{P}$ (which does not appear in $\mathcal{T}$) respectively the extensions $a^{\mathcal{M}'} = o_a$, $b^{\mathcal{M}'} = o_b$, and $\hat{P}^{\mathcal{M}'} = \{(o_a, o_b)\}$. We have that $\mathcal{M}'$ is still a model of $\mathcal{T}$, and moreover $\mathcal{M}' \models \hat{P}(a,b)$, $\mathcal{M}' \models \hat{P} \sqsubseteq E_1$, and $\mathcal{M}' \models \hat{P} \sqsubseteq \neg E_2$. Hence $\mathcal{M}'$ is a model of $\mathcal{K}_{E_1 \sqsubseteq E_2}$, and we get a contradiction. □

THEOREM 24. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{F}$ TBox, R_1 and R_2 two basic roles, $\hat{A}$ an atomic concept not appearing in $\mathcal{T}$, and d a constant. Then,*

1. *$\mathcal{T} \models R_1 \sqsubseteq R_2$ or $\mathcal{T} \models \neg R_1 \sqsubseteq \neg R_2$ if and only if (a) $R_1 = R_2$, or (b) the KB*

$$\mathcal{K}_{\exists R_1 \sqsubseteq \neg \exists R_1} = \langle \mathcal{T} \cup \{\hat{A} \sqsubseteq \exists R_1\},\ \{\hat{A}(d)\}\rangle$$

is unsatisfiable, or (c) the KB

$$\mathcal{K}_{\exists R_1^- \sqsubseteq \neg \exists R_1^-} = \langle \mathcal{T} \cup \{\hat{A} \sqsubseteq \exists R_1^-\},\ \{\hat{A}(d)\}\rangle$$

is unsatisfiable;

2. *$\mathcal{T} \models \neg R_1 \sqsubseteq R_2$ if and only if $\mathcal{T}$ is unsatisfiable.*

3. *$\mathcal{T} \models R_1 \sqsubseteq \neg R_2$ if and only if either (a) the KB*

$$\mathcal{K}_{\exists R_1 \sqsubseteq \neg \exists R_2} = \langle \mathcal{T} \cup \{\hat{A} \sqsubseteq \exists R_1, \hat{A} \sqsubseteq \exists R_2\},\ \{\hat{A}(d)\}\rangle$$

is unsatisfiable, or (b) the KB

$$\mathcal{K}^-_{\exists R_1 \sqsubseteq \neg \exists R_2} = \langle \mathcal{T} \cup \{\hat{A} \sqsubseteq \exists R_1^-, \hat{A} \sqsubseteq \exists R_2^-\},\ \{\hat{A}(d)\}\rangle,$$

is unsatisfiable.

Proof. As for the first item, notice that $\neg R_1 \sqsubseteq \neg R_2$ is equivalent to $R_2 \sqsubseteq R_1$, and therefore we will prove the claim only for subsumption of the latter form. The "if" direction is obvious in the case in which $R_1 = R_2$ or $\mathcal{T}$ is unsatisfiable. As for the case in which $\mathcal{T}$ is satisfiable, $R_1 \neq R_2$, and either $\mathcal{K}_{\exists R_1 \sqsubseteq \neg \exists R_1}$ or $\mathcal{K}_{\exists R_1^- \sqsubseteq \neg \exists R_1^-}$ is unsatisfiable, by Theorem 22 ("if" direction), it follows that either $\mathcal{T} \models \exists R_1 \sqsubseteq \neg \exists R_1$ or $\mathcal{T} \models \exists R_1^- \sqsubseteq \neg \exists R_1^-$, and therefore $\mathcal{T} \models R_1 \sqsubseteq R_2$ (notice that $R_1^\mathcal{I} = \emptyset$ in every interpretation $\mathcal{I}$). As for the "only-if" direction, we assume that $\mathcal{T} \models R_1 \sqsubseteq R_2$, and then construct a model of $\mathcal{T}$ that is not a model of $R_1 \sqsubseteq R_2$, thus obtaining a contradiction. Consider the ABox $\mathcal{A} = \{R_1(a, b)\}$, where a and b are different constants, and the *DL-Lite*$_\mathcal{F}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$. According to Theorem 15, $\mathcal{K}$ is satisfiable if and only if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A}\rangle$. Since $R_1(a, b)$ is the only membership assertion in $\mathcal{A}$, the only NIs that $db(\mathcal{A})$ can violate are $\exists R_1 \sqsubseteq \neg \exists R_1$ and $\exists R_1^- \sqsubseteq \neg \exists R_1^-$, but, by assumption, $\mathcal{T}$ does not imply any of such NIs, and therefore $db(\mathcal{A})$ satisfies $cln(\mathcal{T})$ (indeed, by Lemma 11, for any NI α, if $\mathcal{T} \not\models \alpha$, then $cln(\mathcal{T}) \not\models \alpha$). In particular, by Lemma 14, it follows that $can(\mathcal{K})$ is a model of $\mathcal{K}$, and therefore a model of $\mathcal{T}$. According to the chase rules used in the construction of $chase(\mathcal{K})$, it is easy to see that $R_2(a, b)$ is never added to $chase(\mathcal{K})$, i.e.,

$R_2(a,b) \notin \mathit{chase}(\mathcal{K})$, and therefore, $\mathit{can}(\mathcal{K}) \not\models R_1 \sqsubseteq R_2$, thus leading to a contradiction.

As for the second item, the "if" direction is obvious. As for the "only-if" direction, we consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A} = \emptyset$, i.e., is an empty ABox. Since $\mathcal{T}$ is satisfiable, then $\mathcal{K}$ is satisfiable, and by Lemma 14, it follows that $\mathit{can}(\mathcal{K})$ is a model of $\mathcal{K}$ and therefore a model of $\mathcal{T}$. Since $\mathcal{A} = \emptyset$, then also $\mathit{chase}(\mathcal{K}) = \emptyset$, and therefore $\mathit{can}(\mathcal{K}) \not\models \neg R_1 \sqsubseteq R_2$.

The third item can be proved analogously to the first item. In particular, for the "only-if" direction, in the case in which $\mathcal{T}$ is satisfiable, the proof proceeds in a similar way, by considering an ABox $\mathcal{A} = \{R_1(a,b), R_2(a,b)\}$, where a and b are different constants (it is very easy to see that $\mathit{can}(\langle \mathcal{T}, \mathcal{A} \rangle) \not\models R_1 \sqsubseteq \neg R_2$). □

The following theorem characterizes logical implication of a functionality assertion in *DL-Lite*$_\mathcal{R}$ and *DL-Lite*$_\mathcal{F}$, in terms of subsumption between roles.

THEOREM 25. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ TBox and R a basic role. Then, $\mathcal{T} \models (\mathsf{funct}\ R)$ if and only if either $(\mathsf{funct}\ R) \in \mathcal{T}$ (in the case where $\mathcal{T}$ is a DL-Lite$_\mathcal{F}$ KB), or $\mathcal{T} \models R \sqsubseteq \neg R$.*

Proof. "⇐" The case in which $(\mathsf{funct}\ R) \in \mathcal{T}$ is trivial. Instead, if $\mathcal{T} \models R \sqsubseteq \neg R$, then $R^\mathcal{I} = \emptyset$ and hence $\mathcal{I} \models (\mathsf{funct}\ R)$, for every model $\mathcal{I}$ of $\mathcal{T}$.

"⇒" We assume that neither $(\mathsf{funct}\ R) \in \mathcal{T}$ nor $\mathcal{T} \models R \sqsubseteq \neg R$, and we construct a model of $\mathcal{T}$ that is not a model of $(\mathsf{funct}\ R)$. First of all, notice that, since $\mathcal{T}$ does not imply $R \sqsubseteq \neg R$, it also does not imply $\exists R \sqsubseteq \neg \exists R$ and $\exists R^- \sqsubseteq \neg \exists R^-$. Now, consider the ABox $\mathcal{A} = \{R(a,b), R(a,c)\}$, where a, b, and c are pairwise distinct objects, and the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. According to Theorem 15, $\mathcal{K}$ is satisfiable if and only if $\mathit{db}(\mathcal{A})$ is a model of $\langle \mathit{cln}(\mathcal{T}), \mathcal{A} \rangle$. Since $R(a,b)$ and $R(a,c)$ are the only membership assertions in $\mathcal{A}$, the only assertions that $\mathit{db}(\mathcal{A})$ can violate are (i) the NIs $R \sqsubseteq \neg R$, $\exists R \sqsubseteq \neg \exists R$, and $\exists R^- \sqsubseteq \neg \exists R^-$, and (ii) the functionality assertion $(\mathsf{funct}\ R)$. But, by assumption, $\mathcal{T}$ does not imply any of such assertions, and therefore $\mathit{db}(\mathcal{A})$ satisfies $\mathit{cln}(\mathcal{T})$. In particular, by Lemma 14, it follows that $\mathit{can}(\mathcal{K})$ is a model of $\mathcal{K}$, and therefore a model of $\mathcal{T}$. However, by construction of $\mathcal{A}$, $(\mathsf{funct}\ R)$ is not satisfied in $\mathit{db}(\mathcal{A})$, and hence also not in $\mathit{can}(\mathcal{K})$, which means that $\mathit{can}(\mathcal{K})$ is not a model of $(\mathsf{funct}\ R)$. □

Notice that the role inclusion assertion we are using in Theorem 25 is of the form $\mathcal{T} \models R \sqsubseteq \neg R$, and thus expresses the fact that role R has an empty extension in every model of $\mathcal{T}$. Also, by Theorems 23 and 24,

logical implication of role inclusion assertions can in turn be reduced to knowledge base satisfiability.

Hence, with the above theorems in place, in the following we can concentrate on knowledge base satisfiability only.

3.3. Computational complexity

From the results in the previous subsections we can establish the computational complexity characterization for the classical DL reasoning problems for *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$.

THEOREM 26. *In DL-Lite*$_{\mathcal{R}}$ *and DL-Lite*$_{\mathcal{F}}$*, knowledge base satisfiability is* LogSpace *in the size of the ABox (data complexity) and* PTime *in the size of the whole knowledge base (combined complexity).*

Proof. First, LogSpace data complexity follows directly from FOL-reducibility, since evaluating FOL queries/formulas over a model is LogSpace in the size of the model [30]. As for the combined complexity, we have that $cln(\mathcal{T})$ is polynomially related to the size of the TBox $\mathcal{T}$ and hence q_{unsat} defined above is formed by a number of disjuncts that is polynomial in $\mathcal{T}$. Each disjunct can be evaluated separately and contains either 2 or 3 variables. Now, each disjunct can be evaluated by checking the formula under each of the n^3 possible assignments, where n is the size of the domain of $db(\mathcal{A})$ [30]. Finally, once an assignment is fixed the evaluation of the formula can be done in LogSpace [30]. As a result, we get the PTime bound. □

Taking into account the reductions in Theorems 19, 20, 21, 22, 23, 24, and 25, as a consequence of Lemma 16, we get the following results.

THEOREM 27. *In DL-Lite*$_{\mathcal{R}}$ *and DL-Lite*$_{\mathcal{F}}$*, (concept/role) subsumption and logical implication of functionality assertions are both* PTime *in the size of the TBox, while (concept/role) instance checking is* LogSpace *in the size of the ABox and* PTime *in the size of the whole knowledge base.*

4. Query answering: preliminary properties

In this section we start studying query answering in *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$, and establish some preliminary properties which will be used in the next section.

First, we recall that, in the case where $\mathcal{K}$ is an unsatisfiable KB, the answer to a union of conjunctive queries Q is defined as the finite set of tuples $AllTup(Q, \mathcal{K})$. Therefore, in the following we focus on the case where $\mathcal{K}$ is satisfiable.

We start by showing a central property of the canonical interpretation $can(\mathcal{K})$. In particular, the following lemma shows that, for every model $\mathcal{M}$ of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, there is a homomorphism from $can(\mathcal{K})$ to $\mathcal{M}$ that maps the objects in the extension of concepts and roles in $can(\mathcal{K})$ to objects in the extension of concepts and roles in $\mathcal{M}$.

LEMMA 28. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable DL-Lite$_\mathcal{R}$ or DL-Lite$_\mathcal{F}$ KB, and let $\mathcal{M} = (\Delta^\mathcal{M}, \cdot^\mathcal{M})$ be a model for $\mathcal{K}$. Then, there is a function ψ from $\Delta^{can(\mathcal{K})}$ to $\Delta^\mathcal{M}$ such that:*

(i) for each atomic concept A in $\mathcal{K}$ and each object $o \in \Delta^{can(\mathcal{K})}$, if $o \in A^{can(\mathcal{K})}$ then $\psi(o) \in A^\mathcal{M}$, and

(ii) for each atomic role P in $\mathcal{K}$ and each pair of objects $o, o' \in \Delta^{can(\mathcal{K})}$, if $(o, o') \in P^{can(\mathcal{K})}$ then $(\psi(o), \psi(o')) \in P^\mathcal{M}$.

Proof. We define the function ψ from $\Delta^{can(\mathcal{K})}$ to $\Delta^\mathcal{M}$ by induction on the construction of $chase(\mathcal{K})$, and simultaneously show that properties (i) and (ii) hold.

Base Step. For each constant d occurring in $\mathcal{A}$, we set $\psi(d^{can(\mathcal{K})}) = d^\mathcal{M}$ (notice that each model $\mathcal{M}$ interprets each such constant with an element in $\Delta^\mathcal{M}$). We remember that $chase_0(\mathcal{K}) = \mathcal{A}$, $\Delta^{can_0(\mathcal{K})} = \Delta^{can(\mathcal{K})} = \Gamma_C$, and that, for each constant d occurring in $\mathcal{A}$, $d^{can_0(\mathcal{K})} = d$. Then, it is easy to see that for each object $o_c \in \Delta^{can_0(\mathcal{K})}$ (resp., each pair of objects $o_c^1, o_c^2 \in \Delta^{can_0(\mathcal{K})}$) such that $o_c \in A^{can_0(\mathcal{K})}$, where A is an atomic concept in $\mathcal{K}$ (resp., $(o_c^1, o_c^2) \in P^{can(\mathcal{K})}$, where P is an atomic role in $\mathcal{K}$), we have that $A(o_c) \in chase_0(\mathcal{K})$ (resp., $P(o_c^1, o_c^2) \in chase_0(\mathcal{K})$). Since $\mathcal{M}$ satisfies all membership assertions in $\mathcal{A}$, we also have that $\psi(o_c) \in A^\mathcal{M}$ (resp., $(\psi(o_c^1), \psi(o_c^2)) \in P^\mathcal{M}$)).

Inductive Step. Let us assume that $chase_{i+1}(\mathcal{K})$ is obtained from $chase_i(\mathcal{K})$ by applying rule **cr2**. This means that a PI of the form $A \sqsubseteq \exists R$, where A is an atomic concept in $\mathcal{T}$, and R is a basic role in $\mathcal{T}$, is applied in $chase_i(\mathcal{K})$ to a membership assertion of the form $A(d)$, such that there does not exist a constant $f \in \Gamma_C$ such that $\mathsf{ga}(R, d, f) \in chase_i(\mathcal{K})$. Therefore $chase_{i+1}(\mathcal{K}) = chase_i(\mathcal{K}) \cup \{\mathsf{ga}(R, d, e)\}$, where e follows lexicographically all constants appearing in $chase_i(\mathcal{K})$ (notice that this means that $(d, e) \in R^{can_{i+1}(\mathcal{K})}$). By induction hypothesis, there exists $o_m \in \Delta^\mathcal{M}$ such that that $\psi(d) = o_m$ and $o_m \in A^\mathcal{M}$. Because of the presence of the PI $A \sqsubseteq \exists R$ in $\mathcal{T}$, and because $\mathcal{M}$ is a model of $\mathcal{K}$, there is at least one object $o'_m \in \Delta^\mathcal{M}$ such that $(o_m, o'_m) \in R^\mathcal{M}$. Then, we set $\psi(e) = o'_m$, and we can conclude that $(\psi(d), \psi(e)) \in R^\mathcal{M}$.

With a similar argument we can prove the inductive step also in those cases in which $can_{i+1}(\mathcal{K})$ is obtained from $can_i(\mathcal{K})$ by applying one of the rules **cr1**, **cr3**, **cr4**, or **cr5**, the last one only for *DL-Lite*$_\mathcal{R}$ KBs. □

Based on the above property, we now prove that the canonical model $can(\mathcal{K})$ of a satisfiable KB $\mathcal{K}$ is able to represent all models of $\mathcal{K}$ with respect to unions of conjunctive queries.

THEOREM 29. *Let $\mathcal{K}$ be a satisfiable DL-Lite$_\mathcal{F}$ or DL-Lite$_\mathcal{R}$ KB, and let Q be a union of conjunctive queries over $\mathcal{K}$. Then, $ans(Q, \mathcal{K}) = Q^{can(\mathcal{K})}$.*

Proof. We first recall that $\Delta^{can(\mathcal{K})} = \Gamma_C$ and that, for each constant d occurring in $\mathcal{K}$, we have that $d^{can(\mathcal{K})} = d$. Therefore, given a tuple $\vec{t}$ of constants occurring in $\mathcal{K}$, we have that $\vec{t}^{can(\mathcal{K})} = \vec{t}$. We can hence rephrase the claim as $\vec{t} \in ans(Q, \mathcal{K})$ iff $\vec{t} \in Q^{can(\mathcal{K})}$.

"⇒" Suppose $\vec{t} \in ans(Q, \mathcal{K})$. Then, since $can(\mathcal{K})$ is a model of $\mathcal{K}$, we have that $\vec{t}^{can(\mathcal{K})} \in Q^{can(\mathcal{K})}$.

"⇐" Suppose $\vec{t}^{can(\mathcal{K})} \in Q^{can(\mathcal{K})}$. Let Q be the union of conjunctive queries $Q = \{q_1, \ldots, q_k\}$ with q_i defined as $q_i(\vec{x_i}) \leftarrow conj_i(\vec{x_i}, \vec{y_i})$ for each $i \in \{1, \ldots, k\}$. Then, there exists $i \in \{1, \ldots, k\}$ such that there is a homomorphism from $conj_i(\vec{t}, \vec{y_i})$ to $can(\mathcal{K})$, i.e., there exists an assignment $\mu : V \to \Delta^{can(\mathcal{K})}$ that maps the variables V occurring in $conj_i(\vec{t}, \vec{y_i})$ to objects of $\Delta^{can(\mathcal{K})}$, such that all atoms in $conj_i(\vec{t}, \vec{y_i})$ under the assignment μ evaluate to true in $can(\mathcal{K})$.

Now let $\mathcal{M}$ be a model for $\mathcal{K}$. By Lemma 28, there is a homomorphism ψ from $\Delta^{can(\mathcal{K})}$ to $\Delta^{\mathcal{M}}$. Consequently, the function obtained by composing ψ and μ is a function that maps the variables V occurring in $conj_i(\vec{t}, \vec{y_i})$ to objects of the domain of $\mathcal{M}$, such that all atoms in $conj_i(\vec{t}, \vec{y_i})$ under the assignment μ evaluate to true in $\mathcal{M}$. Therefore, $\vec{t}^{\mathcal{M}} \in Q^{\mathcal{M}}$, which implies that $\vec{t} \in ans(Q, \mathcal{K})$. □

The above property shows that the canonical model $can(\mathcal{K})$ is a correct representative of all the models of a *DL-Lite*$_\mathcal{R}$ (or *DL-Lite*$_\mathcal{F}$) KB with respect to the problem of answering unions of conjunctive queries. In other words, for every union of conjunctive queries Q, the answers to Q over $\mathcal{K}$ correspond to the evaluation of Q in $can(\mathcal{K})$.

In fact, this property holds for all positive FOL queries, but not in general. Consider for example the *DL-Lite*$_{core}$ KB $\mathcal{K} = \langle \emptyset, \{\mathcal{A}_1(d)\} \rangle$, and the FOL boolean query $q = \{ \mid \exists x.A_1(x) \wedge \neg A_2(x)\}$. We have that $chase(\mathcal{K}) = \{\mathcal{A}_1(d)\}$, and therefore q is true in $can(\mathcal{K})$, but the answer to q over $\mathcal{K}$ is false, since there exists a model $\mathcal{M}$ for $\mathcal{K}$ such that q is false in $\mathcal{M}$. Assume, for instance, that $\mathcal{M}$ has the same interpretation

domain as $can(\mathcal{K})$, and that $a^{\mathcal{M}} = a$, $A_1^{\mathcal{M}} = \{a\}$, and $A_2^{\mathcal{M}} = \{a\}$. It is easy to see that $\mathcal{M}$ is a model for $\mathcal{K}$ and q is false in $\mathcal{M}$.

We point out that the canonical interpretation is in general infinite, consequently it cannot be effectively computed in order to solve the query answering problem in *DL-Lite*$_{\mathcal{F}}$ or *DL-Lite*$_{\mathcal{R}}$.

Now, given the limited expressive power of *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$ TBoxes, it might seem that, in order to answer a query over a KB $\mathcal{K}$, we could simply build a *finite* interpretation $\mathcal{I}_{\mathcal{K}}$ that allows for reducing answering *every* union of conjunctive queries (or even every single conjunctive query) over $\mathcal{K}$ to evaluating the query in $\mathcal{I}_{\mathcal{K}}$. The following theorem shows that this is *not* the case.

THEOREM 30. *There exists a DL-Lite*$_{core}$ *KB* $\mathcal{K}$ *for which no finite interpretation* $\mathcal{I}_{\mathcal{K}}$ *exists such that, for every conjunctive query* q *over* $\mathcal{K}$, $ans(q, \mathcal{K}) = q^{\mathcal{I}_{\mathcal{K}}}$.

Proof. Let $\mathcal{K}$ be the *DL-Lite*$_{core}$ KB whose TBox consists of the cyclic concept inclusion $\exists P^- \sqsubseteq \exists P$ and whose ABox consists of the assertion $P(a, b)$.

Let $\mathcal{I}_{\mathcal{K}}$ be a finite interpretation. There are two possible cases:

1. There is no cycle on the relation P in $\mathcal{I}_{\mathcal{K}}$, i.e., the maximum path on the relation $P^{\mathcal{I}_{\mathcal{K}}}$ has a finite length n. In this case, consider the boolean conjunctive query $q \leftarrow P(x_1, x_2), \ldots, P(x_n, x_{n+1})$ that represents the existence of a path of length $n + 1$ in P. It is immediate to verify that the query q is false in $\mathcal{I}_{\mathcal{K}}$, i.e., $q^{\mathcal{I}_{\mathcal{K}}} = \emptyset$, while the answer to q over $\mathcal{K}$ is true, i.e., $ans(q, \mathcal{K}) \neq \emptyset$ (indeed $ans(q, \mathcal{K})$ consists of the empty tuple). This last property can be easily seen by noticing that $q^{can(\mathcal{K})}$ is true.

2. $\mathcal{I}_{\mathcal{K}}$ satisfies the TBox cycle, so it has a finite cycle. More precisely, let us assume that $\mathcal{I}_{\mathcal{K}}$ is such that $(o_1, o_2) \in P^{\mathcal{I}_{\mathcal{K}}}, (o_2, o_3) \in P^{\mathcal{I}_{\mathcal{K}}}, \ldots, (o_n, o_1) \in P^{\mathcal{I}_{\mathcal{K}}}$. In this case, consider the boolean conjunctive query $q \leftarrow P(x_1, x_2), \ldots, P(x_n, x_1)$. It is immediate to verify that such a query is true in $\mathcal{I}_{\mathcal{K}}$, while the answer to q over $\mathcal{K}$ is false. This last property can be easily seen by noticing that $q^{can(\mathcal{K})}$ is false, since $chase(\mathcal{K})$ does not contain a set of facts $P(a_1, a_2), P(a_2, a_3), \ldots, P(a_n, a_1)$, for any n, and therefore in $can(\mathcal{K})$ there does not exist any cycle on the relation P.

Consequently, in both cases $ans(q, \mathcal{K}) \neq q^{\mathcal{I}_{\mathcal{K}}}$. □

The above property demonstrates that answering queries in *DL-Lite*$_{core}$, and hence both in *DL-Lite*$_{\mathcal{R}}$ and in *DL-Lite*$_{\mathcal{F}}$, goes beyond both propositional logic and relational databases.

Finally, we prove a property that relates answering unions of conjunctive queries to answering conjunctive queries.

THEOREM 31. *Let $\mathcal{K}$ be either a DL-Lite$_\mathcal{R}$ or a DL-Lite$_\mathcal{F}$ KB, and let Q be a union of conjunctive queries over $\mathcal{K}$. Then, $ans(Q, \mathcal{K}) = \bigcup_{q_i \in Q} ans(q_i, \mathcal{K})$.*

Proof. The proof that $\bigcup_{q_i \in Q} ans(q_i, \mathcal{K}) \subseteq ans(Q, \mathcal{K})$ is immediate. To prove that $ans(Q, \mathcal{K}) \subseteq \bigcup_{q_i \in Q} ans(q_i, \mathcal{K})$, we distinguish two possible cases:

1. $\mathcal{K}$ is unsatisfiable. Then, it immediately follows that $\bigcup_{q_i \in Q} ans(q_i, \mathcal{K})$ and $ans(Q, \mathcal{K})$ are equal and coincide with the set $AllTup(Q, \mathcal{K})$;

2. $\mathcal{K}$ is satisfiable. In this case, suppose $\vec{t} \in ans(Q, \mathcal{K})$ and suppose that every q_i is of the form $q_i(\vec{x_i}) \leftarrow conj_i(\vec{x_i}, \vec{y_i})$ for each $q_i \in Q$. Then, by Theorem 29, $\vec{t}^{can(\mathcal{K})} \in Q^{can(\mathcal{K})}$, which implies that there exists $i \in \{1, \ldots, k\}$ such that $\vec{t}^{can(\mathcal{K})} \in conj_i(\vec{t}, \vec{y_i})^{can(\mathcal{K})}$. Hence, from Theorem 29, it follows that $\vec{t} \in ans(q_i, \mathcal{K})$.

□

Informally, the above property states that the set of answers to a union of conjunctive queries Q in *DL-Lite*$_\mathcal{R}$ and *DL-Lite*$_\mathcal{F}$ corresponds to the union of the answers to the various conjunctive queries in Q.

5. Query answering in *DL-Lite*$_\mathcal{R}$

In this section we discuss query answering in *DL-Lite*$_\mathcal{R}$. More precisely, based on the properties shown in the previous section, we define an algorithm for answering unions of conjunctive queries in *DL-Lite*$_\mathcal{R}$, and analyze its computational complexity.

In a nutshell, our query answering method strongly separates the intensional and the extensional level of the *DL-Lite*$_\mathcal{R}$ KB: the query is first processed and reformulated based on the TBox axioms; then, the TBox is discarded and the reformulated query is evaluated over the ABox, as if the ABox were a simple relational database (cf. Section 2.5). More precisely, given a query q over $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we compile the assertions of $\mathcal{T}$ (in fact, the PIs in $\mathcal{T}$) into the query itself, thus obtaining a new query q'. Such a new query q' is then evaluated over $db(\mathcal{A})$, thus essentially reducing query answering to query evaluation over a database instance. Since the size of q' does not depend on the

ABox, the data complexity of the whole query answering algorithm is the same as the data complexity of evaluating q'. We show that, in the case where q is a conjunctive query, the query q' is a union of conjunctive queries. Hence, the data complexity of the whole query answering algorithm is polynomial.

In the following, we first define an algorithm for the reformulation of conjunctive queries. Then, we describe a technique for answering unions of conjunctive queries in *DL-Lite*$_\mathcal{R}$ and we prove correctness of such a technique. Finally, we analyze the computational complexity of query answering over *DL-Lite*$_\mathcal{R}$ KBs.

5.1. Query reformulation

We start our presentation by giving some preliminary definitions.

We say that an argument of an atom in a query is *bound* if it corresponds to either a distinguished variable or a shared variable, i.e., a variable occurring at least twice in the query body, or a constant. Instead, an argument of an atom in a query is *unbound* if it corresponds to a non-distinguished non-shared variable. As usual, we use the symbol '$\_$' to represent non-distinguished non-shared variables.

A PI I is *applicable to an atom* $A(x)$, if I has A in its right-hand side.

A PI I is *applicable to an atom* $P(x_1, x_2)$, if: (i) $x_2 = \_$ and the right-hand side of I is $\exists P$; or (ii) $x_1 = \_$ and the right-hand side of I is $\exists P^-$; or (iii) I is a role inclusion assertion and its right-hand side is either P or P^-. Roughly speaking, an inclusion I is applicable to an atom g if the predicate of g is equal to the predicate in the right-hand side of I and, in the case when I is an inclusion assertion between concepts, if g has at most one bound argument and corresponds to the object that is implicitly referred to by the inclusion I.

We indicate with $gr(g, I)$ the atom obtained from the atom g by applying the applicable inclusion I. Formally:

DEFINITION 32. Let I be an inclusion assertion that is applicable to the atom g. Then, $gr(g, I)$ is the atom defined as follows:

- if $g = A(x)$ and $I = A_1 \sqsubseteq A$, then $gr(g, I) = A_1(x)$;
- if $g = A(x)$ and $I = \exists P \sqsubseteq A$, then $gr(g, I) = P(x, \_)$;
- if $g = A(x)$ and $I = \exists P^- \sqsubseteq A$, then $gr(g, I) = P(\_, x)$;
- if $g = P(x, \_)$ and $I = A \sqsubseteq \exists P$, then $gr(g, I) = A(x)$;
- if $g = P(x, \_)$ and $I = \exists P_1 \sqsubseteq \exists P$, then $gr(g, I) = P_1(x, \_)$;
- if $g = P(x, \_)$ and $I = \exists P_1^- \sqsubseteq \exists P$, then $gr(g, I) = P_1(\_, x)$;
- if $g = P(\_, x)$ and $I = A \sqsubseteq \exists P^-$, then $gr(g, I) = A(x)$;
- if $g = P(\_, x)$ and $I = \exists P_1 \sqsubseteq \exists P^-$, then $gr(g, I) = P_1(x, \_)$;

Algorithm PerfectRef $(q, \mathcal{T})$
Input: conjunctive query q, TBox $\mathcal{T}$
Output: union of conjunctive queries PR
$PR := \{q\}$;
repeat
  $PR' := PR$;
  for each $q \in PR'$ **do**
  (a) **for each** g in q **do**
      for each PI I in $\mathcal{T}$ **do**
        if I is applicable to g
        then $PR := PR \cup \{\, q[g/gr(g, I)] \,\}$
  (b) **for each** g_1, g_2 in q **do**
      if g_1 and g_2 unify
      then $PR := PR \cup \{\tau(\mathit{reduce}(q, g_1, g_2))\}$;
until $PR' = PR$;
return PR

Figure 2. The algorithm PerfectRef

- if $g = P(\_, x)$ and $I = \exists P_1^- \sqsubseteq \exists P^-$, then $gr(g, I) = P_1(\_, x)$;
- if $g = P(x_1, x_2)$ and either $I = P_1 \sqsubseteq P$ or $I = P_1^- \sqsubseteq P^-$, then $gr(g, I) = P_1(x_1, x_2)$;
- if $g = P(x_1, x_2)$ and either $I = P_1 \sqsubseteq P^-$ or $P_1^- \sqsubseteq P$, then $gr(g, I) = P_1(x_2, x_1)$.

In Figure 2, we provide the algorithm PerfectRef, which reformulates a conjunctive query taking into account the PIs of a TBox $\mathcal{T}$.

In the algorithm, $q[g/g']$ denotes the conjunctive query obtained from q by replacing the atom g with a new atom g'. Furthermore, τ is a function that takes as input a conjunctive query q and returns a new conjunctive query obtained by replacing each occurrence of an unbound variable in q with the symbol $\_$. Finally, *reduce* is a function that takes as input a conjunctive query q and two atoms g_1 and g_2 occurring in the body of q, and returns a conjunctive query q' obtained by applying to q the *most general unifier* between g_1 and g_2. We point out that, in unifying g_1 and g_2, each occurrence of the $\_$ symbol has to be considered a different unbound variable. The most general unifier substitutes each $\_$ symbol in g_1 with the corresponding argument in g_2, and vice-versa (obviously, if both arguments are $\_$, the resulting argument is $\_$).

Informally, the algorithm first reformulates the atoms of each conjunctive query $q \in PR'$, and produces a new query for each atom reformulation (step (a)). Roughly speaking, PIs are used as rewriting

rules, applied from right to left, which allow one to compile away in the reformulation the intensional knowledge (represented by $\mathcal{T}$) that is relevant for answering q. At step (b), for each pair of atoms g_1, g_2 that unify and occur in the body of a query q, the algorithm computes the conjunctive query $q' = \mathit{reduce}(q, g_1, g_2)$. Thanks to the unification performed by *reduce*, variables that are bound in q may become unbound in q'. Hence, PIs that were not applicable to atoms of q, may become applicable to atoms of q' (in the next executions of step (a)). Notice that the use of τ is necessary in order to guarantee that each unbound variable is represented by the symbol $\_$.

EXAMPLE 33. Consider the query

$$q(x) \leftarrow \mathit{TeachesTo}(x, y), \mathit{TeachesTo}(\_, y)$$

over the TBox of Example 1. In such a query, the atoms $\mathit{TeachesTo}(x, y)$ and $\mathit{TeachesTo}(\_, y)$ unify, and by executing $\mathit{reduce}(q, \mathit{TeachesTo}(x, y), \mathit{TeachesTo}(\_, y))$, we obtain the atom $\mathit{TeachesTo}(x, y)$. The variable y is unbound, and therefore the function τ replaces it with $\_$. Now, the PI $\mathit{Professor} \sqsubseteq \exists \mathit{TeachesTo}$ can be applied to $\mathit{TeachesTo}(x, \_)$, whereas, before the reduction process, it could not be applied to any atom of the query. ∎

The following lemma shows that the algorithm PerfectRef terminates, when applied to a conjunctive query and a *DL-Lite*$_\mathcal{R}$ TBox.

LEMMA 34. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox, and let q be a conjunctive query over $\mathcal{T}$. Then, the algorithm PerfectRef $(q, \mathcal{T})$ terminates.*

Proof. Termination of PerfectRef, for each q and $\mathcal{T}$ in input, immediately follows from the following facts:

1. The maximum number of atoms in the body of a conjunctive query generated by the algorithm is equal to the length of the initial query q. Indeed, in each iteration, a query atom is either replaced with another one, or the number of atoms in the query is reduced; hence, the number of atoms is bounded by the number of atoms in the query. The length of the query is less than or equal to n, where n is the query size, i.e., n is proportional to the number of atoms and the number of terms occurring in the query.

2. The set of terms that occur in the conjunctive queries generated by the algorithm is equal to the set of variables and constants occurring in q plus the symbol $\_$, hence such a set has cardinality less than or equal to $n + 1$, where n is the query size.

3. As a consequence of the above point, the number of different atoms that may occur in a conjunctive query generated by the algorithm is less than or equal to $m \cdot (n+1)^2$, where m is the number of predicate symbols (concept or role names) that occur either in the TBox or in the query.

4. The algorithm does not drop queries that it has generated.

The above points 1 and 3 imply that the number of distinct conjunctive queries generated by the algorithm is finite, whereas point 4 implies that the algorithm does not generate a query more than once, and therefore PerfectRef terminates. Precisely, the number of distinct conjunctive queries generated by the algorithm is less than or equal to $(m \cdot (n+1)^2)^n$, which corresponds to the maximum number of executions of the repeat–until cycle of the algorithm. □

EXAMPLE 35. Referring to the *DL-Lite*$_{core}$ TBox $\mathcal{T}$ in Example 1, consider the conjunctive query q:

$$q(x) \leftarrow \mathit{TeachesTo}(x, y), \mathit{HasTutor}(y, \_)$$

asking for professors that teach to students that have a tutor.

Let us analyze the execution of the algorithm PerfectRef$(q, \mathcal{T})$. At the first execution of step (a), the algorithm inserts in PR the new query

$$q(x) \leftarrow \mathit{TeachesTo}(x, y), \mathit{Student}(y)$$

by applying to the atom $\mathit{HasTutor}(y, \_)$ the PI $\mathit{Student} \sqsubseteq \exists \mathit{HasTutor}$. Then, at a second execution of step (a), the query

$$q(x) \leftarrow \mathit{TeachesTo}(x, y), \mathit{TeachesTo}(\_, y)$$

is added to PR, according to application of the PI $\exists \mathit{TeachesTo}^- \sqsubseteq \mathit{Student}$ to the atom $\mathit{Student}(y)$. Since the two atoms of the second query unify, step (b) of the algorithm inserts the query

$$q(x) \leftarrow \mathit{TeachesTo}(x, \_)$$

into PR. Notice that the variable y is unbound in the new query, hence it has been replaced by the symbol _. At a next iteration, step (a) produces the query

$$q(x) \leftarrow \mathit{Professor}(x)$$

by applying $\mathit{Professor} \sqsubseteq \exists \mathit{TeachesTo}$ to $\mathit{TeachesTo}(x, \_)$, and then, at a further execution of step (a), it generates the query

$$q(x) \leftarrow \mathit{HasTutor}(\_, x)$$

by applying $\exists HasTutor^- \sqsubseteq Professor$ to $Professor(x)$. The set constituted by the above five queries and the original query q is then returned by the algorithm. ∎

EXAMPLE 36. As a further example, consider now the *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}'$ obtained from $\mathcal{T}$ by adding the inclusion assertion $HasTutor^- \sqsubseteq TeachesTo$, and the conjunctive query q' defined as follows:

$$q'(x) \leftarrow Student(x)$$

Then, the result of $\mathsf{PerfectRef}(q', \mathcal{T}')$ is the union of:

$$\begin{array}{l} q'(x) \leftarrow Student(x) \\ q'(x) \leftarrow TeachesTo(\_, x) \\ q'(x) \leftarrow HasTutor(x, \_) \end{array}$$

Notice that, without considering the new inclusion assertion between roles, we would have obtained only the union of the first two conjunctive queries as result of the algorithm $\mathsf{PerfectRef}(q', \mathcal{T})$. ∎

We note that the union of conjunctive queries produced by PerfectRef is not necessarily minimal, i.e., it may contain pairs of conjunctive queries that are one contained into the other. Though this does not affect the worst-case computational complexity, for practical purposes this set of queries can be simplified, using well-known minimization techniques for relational queries.

5.2. Query evaluation

In order to compute the answers to q over the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we need to evaluate the set of conjunctive queries PR produced by the algorithm PerfectRef over the ABox $\mathcal{A}$ considered as a relational database.

In Figure 3, we define the algorithm Answer that, given a KB $\mathcal{K}$ and a union of conjunctive queries Q of arity n, computes $ans(Q, \mathcal{K})$. The following theorem shows that the algorithm Answer terminates, when applied to a union of conjunctive queries and a *DL-Lite*$_\mathcal{R}$ TBox.

THEOREM 37. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ KB, let Q be a union of conjunctive queries. Then, the algorithm $\mathsf{Answer}(Q, \mathcal{K})$ terminates.*

Proof. Termination of $\mathsf{Answer}(Q, \mathcal{K})$ follows straightforwardly from Lemma 16 and Lemma 34, which respectively establish termination of algorithms $\mathsf{Consistent}(\mathcal{K})$ and $\mathsf{PerfectRef}\ (q, \mathcal{T})$, for each conjunctive query $q \in Q$. □

Algorithm $\mathsf{Answer}(Q, \mathcal{K})$
Input: UCQ Q, KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: $ans(Q, \mathcal{K})$
if not $\mathsf{Consistent}(\mathcal{K})$
then return $AllTup(Q, \mathcal{K})$
else return $(\bigcup_{q_i \in Q} \mathsf{PerfectRef}(q_i, \mathcal{T}))^{db(\mathcal{A})}$;

Figure 3. The algorithm Answer

EXAMPLE 38. Let us consider again the query of Example 35

$$q(x) \leftarrow TeachesTo(x, y), HasTutor(y, \_)$$

expressed over the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A}$ contains the assertions:

$$Student(\mathsf{John}), \quad HasTutor(\mathsf{John}, \mathsf{Mary}), \quad TeachesTo(\mathsf{Mary}, \mathsf{Bill}).$$

By executing $\mathsf{Answer}(q, \mathcal{K})$, since $\mathcal{K}$ is satisfiable (see Section 3), it executes $\mathsf{PerfectRef}(q, \mathcal{T})$, which returns the union of conjunctive queries described in Example 35. Let Q be such a query, then it is easy to see that $Q^{db(\mathcal{A})}$ is the set $\{\mathsf{Mary}\}$.

Let us now consider the query

$$q'(x) \leftarrow Student(x),$$

expressed over the KB $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$, where $\mathcal{T}'$ is as in Example 36, and $\mathcal{A}'$ contains the assertions

$$HasTutor(\mathsf{John}, \mathsf{Mary}), \quad TeachesTo(\mathsf{Mary}, \mathsf{Bill}).$$

Obviously, $\mathcal{K}'$ is satisfiable, and by executing $\mathsf{Answer}(q', \mathcal{K}')$ we obtain the answer set $\{\mathsf{John}, \mathsf{Bill}\}$ by the evaluation of the union of conjunctive queries returned by $\mathsf{PerfectRef}(q', \mathcal{T}')$, and which we have described in Example 36. Notice that, without considering the new inclusion assertion between roles, we would have obtained only $\{\mathsf{Bill}\}$ as answer to the query. ∎

5.3. Correctness

We now prove correctness of the above described query answering technique. As discussed in the previous section, from Theorem 29 it follows that query answering can in principle be done by evaluating the query over the model $can(\mathcal{K})$. However, since $can(\mathcal{K})$ is in general infinite, we

obviously avoid the construction of $can(\mathcal{K})$. Rather, as we said before, we are able to compile the TBox into the query, thus simulating the evaluation of the query over $can(\mathcal{K})$ by evaluating a finite reformulation of the query over the ABox considered as a database.

LEMMA 39. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox, q a conjunctive query over $\mathcal{T}$, and PR the union of conjunctive queries returned by* PerfectRef$(q, \mathcal{T})$. *For every DL-Lite$_\mathcal{R}$ ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, $ans(q, \langle \mathcal{T}, \mathcal{A} \rangle) = PR^{db(\mathcal{A})}$.*

Proof. We first introduce the preliminary notion of witness of a tuple of constants with respect to a conjunctive query. Given a *DL-Lite*$_\mathcal{R}$ knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a conjunctive query $q(\vec{x}) \leftarrow conj(\vec{x}, \vec{y})$ over $\mathcal{K}$, and a tuple $\vec{t}$ of constants occurring in $\mathcal{K}$, a set of membership assertions $\mathcal{G}$ is a *witness of $\vec{t}$ w.r.t. q* if there exists a substitution σ from the variables $\vec{y}$ in $conj(\vec{t}, \vec{y})$ to constants in $\mathcal{G}$ such that the set of atoms in $\sigma(conj(\vec{t}, \vec{y}))$ is equal to $\mathcal{G}$. In particular, we are interested in witnesses of a tuple $\vec{t}$ w.r.t. a query q that are contained in $chase(\mathcal{K})$. Intuitively, each such witness corresponds to a subset of $chase(\mathcal{K})$ that is sufficient in order to have that the formula $\exists \vec{y}.conj(\vec{t}, \vec{y})$ evaluates to true in the canonical intepretation $can(\mathcal{K})$, and therefore the tuple $\vec{t} = \vec{t}^{can(\mathcal{K})}$ belongs to $q^{can(\mathcal{K})}$. More precisely, we have that $\vec{t} \in q^{can(\mathcal{K})}$ iff there exists a witness $\mathcal{G}$ of $\vec{t}$ w.r.t. q such that $\mathcal{G} \subseteq chase(\mathcal{K})$. The cardinality of a witness $\mathcal{G}$, denoted by $|\mathcal{G}|$, is the number of membership assertions in $\mathcal{G}$.

Since $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, by Theorem 29, $ans(q, \langle \mathcal{T}, \mathcal{A} \rangle) = q^{can(\mathcal{K})}$, and, by Theorem 31, $PR^{db(\mathcal{A})} = \bigcup_{\hat{q} \in PR} \hat{q}^{db(\mathcal{A})}$, where PR is the union of conjunctive queries returned by PerfectRef$(q, \mathcal{T})$. Consequently, to prove the claim it is sufficient to show that $\bigcup_{\hat{q} \in PR} \hat{q}^{db(\mathcal{A})} = q^{can(\mathcal{K})}$.

"$\Leftarrow$" We have to prove that $\hat{q}^{db(\mathcal{A})} \subseteq q^{can(\mathcal{K})}$, for each $\hat{q} \in PR$. We initially consider the conjunctive queries q_i and q_{i+1}, such that q_{i+1} is obtained from q_i by means of step (a) of the algorithm PerfectRef, and show that $q_{i+1}^{can(\mathcal{K})} \subseteq q_i^{can(\mathcal{K})}$. Let $\vec{t}$ be a tuple of constants occurring in $\mathcal{K}$ such that $\vec{t}^{can(\mathcal{K})} \in q_{i+1}^{can(\mathcal{K})}$. Then, it follows that there exists $\mathcal{G} \subseteq can(\mathcal{K})$ such that $\mathcal{G}$ is a witness of $\vec{t}$ w.r.t. q_{i+1}. Let us assume that q_{i+1} is obtained from q_i by applying step (a) when the positive inclusion assertion I of $\mathcal{T}$ is of the form $A_1 \sqsubseteq A$, i.e., $q_{i+1} = q_i[A(x)/A_1(x)]$ (the proof when I is of the other forms listed in Definition 32 is analogous). Then, there exists a membership assertion in $\mathcal{G}$ to which the PI $A_1 \sqsubseteq A$ is applicable, which implies that there exists a witness of $\vec{t}$ w.r.t. q_i contained in $chase(\mathcal{K})$. Therefore, $\vec{t}^{can(\mathcal{K})} \in q_i^{can(\mathcal{K})}$. Consider now the case in which q_{i+1} is obtained from q_i by applying step (b) of the algorithm,

i.e., $q_{i+1} = \tau(reduce(q_i, g_1, g_2))$, where g_1, g_2 are two atoms belonging to q_i such that g_1 and g_2 unify. It is easy to see that in such a case $\mathcal{G}$ is also a witness of $\vec{t}$ w.r.t. q_i, and therefore $\vec{t}^{can(\mathcal{K})} \in q_i^{can(\mathcal{K})}$. Since each query of PR is either q or a query obtained from q by repeatedly applying steps (a) and (b) of the algorithm PerfectRef, it follows that for each $\hat{q} \in PR$, $\hat{q}^{can(\mathcal{K})} \subseteq q^{can(\mathcal{K})}$, by repeatedly applying the property $(q_{i+1})^{can(\mathcal{K})} \subseteq (q_i)^{can(\mathcal{K})}$. Thus, we have shown that $\hat{q}^{db(\mathcal{A})} \subseteq \hat{q}^{can(\mathcal{K})}$ for each conjunctive query $\hat{q} \in PR$.

"$\Rightarrow$" We have to show that for each tuple $\vec{t} \in q^{can(\mathcal{K})}$, there exists $\hat{q} \in PR$ such that $\vec{t} \in \hat{q}^{db(\mathcal{A})}$. First, since $\vec{t} \in q^{can(\mathcal{K})}$, it follows that there exists a finite number k such that there is a witness $\mathcal{G}_k$ of $\vec{t}$ w.r.t. q contained in $chase_k(\mathcal{K})$. Moreover, without loss of generality, we can assume that every rule **cr1**, **cr2**, **cr3**, **cr4**, and **cr5** used in the construction of $chase(\mathcal{K})$ is necessary in order to generate such a witness $\mathcal{G}_k$: i.e., $chase_k(\mathcal{K})$ can be seen as a forest (set of trees) where: (i) the roots correspond to the membership assertions of $\mathcal{A}$; (ii) there are exactly k edges, where each edge corresponds to an application of a rule; (iii) each leaf is either also a root or a membership assertion in $\mathcal{G}_k$. In the following, we say that a membership assertion f is an ancestor of a membership assertion f' in a set of membership assertions $\mathcal{S}$, if there exist $f_1, \ldots, f_n$ in $\mathcal{S}$, where $f_1 = f$ and $f_n = f'$, such that, for each $2 \leq i \leq n$, f_i can be generated by applying a chase rule to f_{i-1}. We also say that f' is a successor of f. Furthermore, for each $i \in \{0, \ldots, k\}$, we denote with $\mathcal{G}_i$ the *pre-witness* of $\vec{t}$ w.r.t. q in $chase_k(\mathcal{K})$, defined as follows:

$$\begin{aligned}\mathcal{G}_i = \{\ f \in chase_i(\mathcal{K}) \mid &\text{there exists } f' \in \mathcal{G}_k \text{ s.t. } f \text{ is an ancestor of } f' \\ &\text{in } chase_k(\mathcal{K}) \text{ and there exists no successor of } f \text{ in } chase_i(\mathcal{K}) \\ &\text{that is an ancestor of } f' \text{ in } chase_k(\mathcal{K})\ \}\end{aligned}$$

Now we prove by induction on i that, starting from $\mathcal{G}_k$, we can "go back" through the rule applications and find a query $\hat{q}$ in PR such that the pre-witness $\mathcal{G}_{k-i}$ of $\vec{t}$ w.r.t. q in $chase_{k-i}(\mathcal{K})$ is also a witness of $\vec{t}$ w.r.t. $\hat{q}$. To this aim, we prove that there exists $\hat{q} \in PR$ such that $\mathcal{G}_{k-i}$ is a witness of $\vec{t}$ w.r.t. $\hat{q}$ and $|\hat{q}| = |\mathcal{G}_{k-i}|$, where $|\hat{q}|$ indicates the number of atoms in the conjunctive query $\hat{q}$. The claim then follows for $i = k$, since $chase_0(\mathcal{K}) = \mathcal{A}$.

Base step: There exists $\hat{q} \in PR$ such that $\mathcal{G}_k$ is a witness of $\vec{t}$ w.r.t. $\hat{q}$ and $|\hat{q}| = |\mathcal{G}_k|$. This is an immediate consequence of the membership assertions that: (i) $q \in PR$; (ii) PR is closed with respect to step (b) of the algorithm PerfectRef. Indeed, if $|\mathcal{G}_k| < |q|$ then there exist two atoms g_1, g_2 in q and a membership assertion f in $\mathcal{G}_k$ such that f and g_1 unify and f and g_2 unify, which implies that g_1 and g_2 unify. Therefore,

by step (b) of the algorithm, it follows that there exists a query $q_1 \in PR$ (with $q_1 = \mathit{reduce}(q, g_1, g_2)$) such that $\mathcal{G}_k$ is a witness of $\vec{t}$ w.r.t. q_1 and $|q_1| = |q| - 1$. Now, if $|\mathcal{G}_k| < |q_1|$, we can iterate the above argument, thus we conclude that there exists $\hat{q} \in PR$ such that $\mathcal{G}_k$ is a witness of $\vec{t}$ w.r.t. $\hat{q}$ and $|\hat{q}| = |\mathcal{G}_k|$.

Inductive step: suppose that there exists $\hat{q} \in PR$ such that $\mathcal{G}_{k-i+1}$ is a witness of $\vec{t}$ w.r.t. $\hat{q}$ and $|\hat{q}| = |\mathcal{G}_{k-i+1}|$. Let us assume that $\mathit{chase}_{k-i+1}(\mathcal{K})$ is obtained by applying **cr2** to $\mathit{chase}_{k-i}(\mathcal{K})$ (the proof is analogous for rules **cr1**, **cr3**, **cr4**, and **cr5**). This means that a PI of the form $A \sqsubseteq \exists P$[8], where A is an atomic concept and P is an atomic role, is applied in $\mathit{chase}_{k-i}(\mathcal{K})$ to a membership assertion of the form $A(a)$, such that there does not exists $d \in \Gamma_C$ such that $P(a, d) \in \mathit{chase}_{k-i}(\mathcal{K})$. Therefore, $\mathit{chase}_{k-i+1}(\mathcal{K}) = \mathit{chase}_{k-i}(\mathcal{K}) \cup P(a, a_n)$, where $a_n \in \Gamma_C$ follows lexicographically all constants occurring in $\mathit{chase}_i(\mathcal{K})$.

Since a_n is a new constant of Γ_C, i.e., a constant not occurring elsewhere in $\mathcal{G}_{k-i+1}$, and since $|\hat{q}| = |\mathcal{G}_{k-i+1}|$, it follows that the atom $P(x, \_)$ occurs in $\hat{q}$. Therefore, by step (a) of the algorithm, it follows that there exists a query $q_1 \in PR$ (with $q_1 = \hat{q}[P(x, \_)/A(x)]$) such that $\mathcal{G}_{k-i}$ is a witness of $\vec{t}$ w.r.t. q_1.

Now, there are two possible cases: either $|q_1| = |\mathcal{G}_{k-i}|$, and in this case the claim is immediate; or $|q_1| = |\mathcal{G}_{k-i}| + 1$. This last case arises if and only if the membership assertion $A(a)$ to which the rule **cr2** is applied is both in $\mathcal{G}_{k-i}$ and in $\mathcal{G}_{k-i+1}$. This implies that there exist two atoms g_1 and g_2 in q_1 such that $A(a)$ and g_1 unify and $A(a)$ and g_2 unify, hence g_1 and g_2 unify. Therefore, by step (b) of the algorithm (applied to q_1), it follows that there exists $q_2 \in PR$ (with $q_2 = \mathit{reduce}(q_1, g_1, g_2)$) such that $\mathcal{G}_{k-i}$ is a witness of $\vec{t}$ w.r.t. q_2 and $|q_2| = |\mathcal{G}_{k-i+1}|$, which proves the claim. □

Based on the above property, we are finally able to establish correctness of the algorithm Answer.

THEOREM 40. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ KB, Q a union of conjunctive queries, and $\vec{t}$ a tuple of constants in $\mathcal{K}$. Then, $\vec{t} \in \mathit{ans}(Q, \mathcal{K})$ if and only if $\vec{t} \in \mathsf{Answer}(Q, \mathcal{K})$.*

Proof. In the case where $\mathcal{K}$ is satisfiable, the proof follows immediately from Lemma 39 and Theorem 31. In the case where $\mathcal{K}$ is not satisfiable, it is immediate to verify that the set $\mathit{AllTup}(Q, \mathcal{K})$ returned by $\mathsf{Answer}(Q, \mathcal{K})$ corresponds to $\mathit{ans}(Q, \mathcal{K})$, according to the semantics of queries given in Section 2. □

[8] The other execution of **cr2** is the case when $I = A \sqsubseteq \exists P^-$, which is analogous.

As an immediate corollary of the above properties, it follows that the problem of answering unions of conjunctive queries over satisfiable *DL-Lite*$_\mathcal{R}$ KBs is FOL-reducible. Moreover, it is easy to see that such a FOL-reducibility also extends to the case of arbitrary (both satisfiable and unsatisfiable) *DL-Lite*$_\mathcal{R}$ KBs. Indeed, the whole query answering task can be encoded into a single union of conjunctive queries, obtained by adding to the query $\bigcup_{q_i \in Q} \mathsf{PerfectRef}(q_i, \mathcal{T}))$ a finite number of conjunctions encoding the fact that every tuple in $\mathit{AllTup}(Q, \mathcal{K})$ is in the answer set of the query if $\mathcal{K}$ is unsatisfiable. (For details on the construction of such a query see e.g. [10], which defines an analogous encoding in the context of relational database integrity constraints.) We therefore get the following theorem.

THEOREM 41. *Answering unions of conjunctive queries in DL-Lite*$_\mathcal{R}$ *is FOL-reducible.*

5.4. Computational complexity

We first establish complexity of the algorithm PerfectRef.

LEMMA 42. *Let $\mathcal{T}$ be a DL-Lite*$_\mathcal{R}$ *TBox, and q a conjunctive query over $\mathcal{T}$. The algorithm PerfectRef $(q, \mathcal{T})$ runs in time polynomial in the size of $\mathcal{T}$.*

Proof. Let n be the query size, and let m be the number of predicate symbols (concept or role names) that occur either in the TBox or in the query. As shown in Lemma 34, the number of distinct conjunctive queries generated by the algorithm is less than or equal to $(m \cdot (n+1)^2)^n$, which corresponds to the maximum number of executions of the repeat–until cycle of the algorithm. Since m is linearly bound by the size of the TBox $\mathcal{T}$, while n does not depend on the size of $\mathcal{T}$, from the above point it follows that the algorithm PerfectRef $(q, \mathcal{T})$ runs in time polynomial in the size of $\mathcal{T}$. □

Based on the above property, we are able to establish the complexity of answering unions of conjunctive queries in *DL-Lite*$_\mathcal{R}$.

THEOREM 43. *Answering unions of conjunctive queries in DL-Lite*$_\mathcal{R}$ *is* PTime *in the size of the TBox, and* LogSpace *in the size of the ABox (data complexity).*

Proof. The proof is an immediate consequence of the correctness of the algorithm Answer for *DL-Lite*$_\mathcal{R}$, established in Theorem 40, and the following facts: (*i*) Lemma 42, which implies that the query

$\bigcup_{q_i \in Q} \mathsf{PerfectRef}(q_i, \mathcal{T})$ can be computed in time polynomial in the size of the TBox and constant in the size of the ABox (data complexity); (*ii*) Theorem 26, which states the computational complexity of checking satisfiability of *DL-Lite*$_\mathcal{R}$ KBs; (*iii*) the fact that the evaluation of a union of conjunctive queries over a database can be computed in LogSpace with respect to the size of the database (since unions of conjunctive queries are a subclass of FOL queries). □

We are also able to characterize the combined complexity (i.e., the complexity w.r.t. the size of $\mathcal{K}$ and Q) of answering unions of conjunctive queries in *DL-Lite*$_\mathcal{R}$.

THEOREM 44. *Answering unions of conjunctive queries in DL-Lite*$_\mathcal{R}$ *is NP-complete in combined complexity.*

Proof. To prove membership in NP, observe that a version of the algorithm PerfectRef that nondeterministically returns only one of the conjunctive queries belonging to the reformulation of the input query, runs in nondeterministic polynomial time in combined complexity, since every query returned by PerfectRef can be generated after a polynomial number of transformations of the initial conjunctive query (i.e., after a polynomial number of executions of steps (a) and (b) of the algorithm). This allows the corresponding nondeterministic version of the algorithm Answer to run in nondeterministic polynomial time when the input is a boolean query. NP-hardness follows from NP-hardness of conjunctive query evaluation over relational databases. □

Summarizing, the above results show a very nice computational behavior of queries in *DL-Lite*$_\mathcal{R}$: reasoning in *DL-Lite*$_\mathcal{R}$ is computationally no worse than standard conjunctive query answering (and containment) in relational databases.

6. Query answering in *DL-Lite*$_\mathcal{F}$

In this section, we discuss query answering in *DL-Lite*$_\mathcal{F}$, and analyze its computational complexity. In a nutshell, the technique for query answering closely resembles that for *DL-Lite*$_\mathcal{R}$, hence, it is also based on reformulating the query based on the TBox assertions. The differences with respect to the case of *DL-Lite*$_\mathcal{R}$ are the following. (*i*) On the one hand, the PIs that can appear in a *DL-Lite*$_\mathcal{F}$ TBox are just a subset of those allowed for a *DL-Lite*$_\mathcal{R}$ TBox, namely inclusions of a basic concept in a concept, but no role inclusions. As a consequence, the reformulation rules that can be applied to the atoms of the query are just a subset of those for *DL-Lite*$_\mathcal{R}$ in Definition 32. (*ii*) On the

other hand, a *DL-Lite*$_{\mathcal{F}}$ TBox may contain functionality assertions, which may interact with the inclusion assertions in the TBox. However, this interaction is only of a limited form. Indeed, as already shown in Lemma 8, if a functionality assertion (funct R) is satisfied in the interpretation $db(\mathcal{A})$ corresponding to the ABox $\mathcal{A}$ of a KB $\mathcal{K}$, then (funct R) is also satisfied in the canonical interpretation $can(\mathcal{K})$. Hence, (the simplified form of) the reformulation technique of *DL-Lite*$_{\mathcal{R}}$ can also be applied to *DL-Lite*$_{\mathcal{F}}$ query answering, provided that we take into account also functionality assertions when we check satisfiability of the knowledge base. We now show this formally.

Given a *DL-Lite*$_{\mathcal{F}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a conjunctive query q, the reformulation PR of q is computed, just as for *DL-Lite*$_{\mathcal{R}}$, by the algorithm PerfectRef, considering that inclusion assertions between roles are not present in $\mathcal{T}$, and hence the rules in Definition 32 for such inclusions assertions will never be applied.

Similarly, in order to compute the answer $ans(Q, \mathcal{K})$ to the union of conjunctive queries Q over a *DL-Lite*$_{\mathcal{F}}$ KB $\mathcal{K}$, we can simply invoke Answer on Q and $\mathcal{K}$, noticing that now the satisfiability check takes into account also functionality assertions in $\mathcal{K}$. It is easy to see that also in the case when $\mathcal{K}$ is a *DL-Lite*$_{\mathcal{F}}$ KB the algorithm Answer$(Q, \mathcal{K})$ terminates (cf. Theorem 37).

The following theorem, which follows easily from the already proved results, establishes that this way of proceeding is indeed correct.

THEOREM 45. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{\mathcal{F}}$ KB, Q a union of conjunctive queries, and $\vec{t}$ a tuple of constants in $\mathcal{K}$. Then, $\vec{t} \in ans(Q, \mathcal{K})$ if and only if $\vec{t} \in$ Answer$(Q, \mathcal{K})$.*

Proof. In the case where $\mathcal{K}$ is not satisfiable, as for Theorem 40, it is immediate to verify that the set $AllTup(Q, \mathcal{K})$ returned by Answer$(Q, \mathcal{K})$ corresponds to $ans(Q, \mathcal{K})$, according to the semantics of queries given in Section 2.

So, let us consider the case where $\mathcal{K}$ is satisfiable. Let $\mathcal{T}_I$ be the set of inclusion assertions in $\mathcal{T}$, and let $\mathcal{K}_I = \langle \mathcal{T}_I, \mathcal{A} \rangle$. Note that $\mathcal{K}_I$ is the *DL-Lite*$_{\mathcal{R}}$ KB obtained from $\mathcal{K}$ by removing all functionality assertions in $\mathcal{T}$. Let q be any conjunctive query. Since, by Definition 5, functionality assertions do not play any role in the construction of $chase(\mathcal{K})$, we have that $can(\mathcal{K}) = can(\mathcal{K}_I)$, and hence $q^{can(\mathcal{K})} = q^{can(\mathcal{K}_I)}$. Since $\mathcal{K}$ is satisfiable also $\mathcal{K}_I$ is satisfiable. Hence, by applying Theorem 29 to both $\mathcal{K}$ and $\mathcal{K}_I$, we get that $ans(q, \mathcal{K}) = q^{can(\mathcal{K})} = q^{can(\mathcal{K}_I)} = ans(q, \mathcal{K}_I)$. Let PR be the union of the conjunctive queries returned by PerfectRef$(q, \mathcal{T}_I)$. Since $\mathcal{K}_I = \langle \mathcal{T}_I, \mathcal{A} \rangle$ is a *DL-Lite*$_{\mathcal{R}}$ KB, by Lemma 39,

we have that $ans(q, \langle \mathcal{T}_I, \mathcal{A} \rangle) = PR^{db(\mathcal{A})}$, and hence also $ans(q, \mathcal{K}) = PR^{db(\mathcal{A})}$.

By Theorem 31, and since $\mathsf{Answer}(Q, \mathcal{K})$ computes precisely the union over all $q_i \in Q$ of the evaluations of $\mathsf{PerfectRef}(q_i, \mathcal{T}_I)$ on $db(\mathcal{A})$, we get the claim. □

As an immediate consequence of the theorem above, we get the following property, which is the analogous of Theorem 41, given for *DL-Lite*$_\mathcal{R}$.

THEOREM 46. *Answering unions of conjunctive queries in DL-Lite*$_\mathcal{F}$ *is FOL-reducible.*

As for computational complexity we get the same bounds as those shown in Section 5.

THEOREM 47. *Answering unions of conjunctive queries in DL-Lite*$_\mathcal{F}$ *is* PTIME *in the size of the TBox,* LOGSPACE *in the size of the ABox (data complexity), and NP-complete in combined complexity.*

7. Related work

The DLs of the *DL-Lite* family are fragments of expressive DLs with assertions and inverses studied in the 90's (see [5] for an overview), which are at the base of current ontology languages such as OWL, and for which optimized automated reasoning systems such as Fact and Racer have been developed. Indeed, one could use, off-the-shelf, a system like Racer to perform KB satisfiability, instance checking, and subsumption in the DLs of the *DL-Lite* family. Also, reasoning with conjunctive queries in these DLs has been studied (see e.g., [16]), although not yet implemented in systems. Unfortunately, the reasoning procedures for these DLs are all EXPTIME-hard, and more importantly they are not tailored towards obtaining tight complexity bounds with respect to data complexity. Conjunctive queries combined with DLs were also considered in [20], but again data complexity was not the main concern.

Alternative reasoning procedures that allow for clearly isolating data complexity have recently been proposed, but how they will work in practice still needs to be understood: in [24], a coNP upper bound for data complexity of instance checking in an expressive DL has been shown, and a polynomial fragment has been isolated, though it is open whether the technique can be extended to deal efficiently with

conjunctive queries; building on the technique proposed in [26], coNP-completeness of answering conjunctive queries for an expressive DL with assertions, inverse roles, and number restrictions (that generalize functionality) has been shown in [27].

We observe that *DL-Lite*$_{\mathcal{R}}$ can also capture (the DL-subset of) RDFS[9]. In fact, the query answering technique for *DL-Lite*$_{\mathcal{R}}$ works also for full RDFS extended with participation constraints (i.e., inclusion assertions with $\exists R$ on the right-hand side), and one can show that in this case query answering is indeed LogSpace. However, if we further extend RDFS with functionality assertions, it can be shown that query answering becomes NLogSpace-hard [14]. Finally, if we move from RDFS to DLP [23], query answering becomes PTime-hard, as shown in [14].

There has been a lot of work in DLs on the boundary between polynomial and exponential reasoning. Such a work first concentrated on DLs without the TBox component of the knowledge base, and led to the development of simple DLs, such as $\mathcal{ALN}$, that admit polynomial instance checking. However, for minor variants of $\mathcal{ALN}$, such as $\mathcal{ALE}$ (where qualified existential is introduced and number restrictions are dropped), $\mathcal{FLE}^-$ (where additionally negated atomic concepts are dropped), and $\mathcal{ALU}$ (where union is introduced and number restrictions are dropped), instance checking, and therefore conjunctive query answering, is coNP-complete in data complexity [19]. Indeed, the argument used in the proof of coNP-hardness of $\mathcal{ALE}$, $\mathcal{FLE}^-$, and $\mathcal{ALU}$ in [19], immediately implies the following theorem.

THEOREM 48. *Answering conjunctive queries is coNP-hard in data complexity, if we extend DL-Lite*$_{core}$ *with one of the following features: (1) either* $\forall R.A$ *or* $\neg A$ *can appear in the left-hand side of inclusion assertions; (2) either* $\forall R.A$ *or* $\neg A$ *can appear as atoms in the query; (3) union of concepts can appear in the right-hand side of inclusion assertions.*

If we allow for cyclic inclusion assertions in the TBox, then even subsumption in CLASSIC and $\mathcal{ALN}$ becomes intractable [11][10]. Observe that the DLs of the *DL-Lite* family do allow for cyclic assertions without falling into intractability. Indeed, we can enforce the cyclic propagation of the existence of an R-successor using the two inclusion assertions $A \sqsubseteq \exists R$ and $\exists R^- \sqsubseteq A$. The constraint imposed on a model by such assertions is similar to the one imposed by the $\mathcal{ALN}$ cyclic assertion $A \sqsubseteq \exists R \sqcap \forall R.A$, though stronger, since it additionally enforces the

[9] `http://www.w3.org/TR/rdf-schema/`

[10] Note that a TBox with only acyclic inclusion assertions can always be transformed into an empty TBox.

second component of R to be typed by A. In order to keep tractability even in the presence of cycles, the *DL-Lite* family imposes restrictions on the use of the $\forall R.C$ construct, which, if used together with inclusion assertions, immediately would lead to intractability [11].

More recently languages equipped with *qualified existential restrictions* but no universal restrictions (even expressed in a disguised way, as in *DL-Lite*, through inverse roles) have been studied. In particular, in [4] it has been shown that for the basic language $\mathcal{EL}$ instance checking is polynomial even in the presence of general (i.e., cyclic) inclusion assertions, while extensions of the language lead easily to intractability. Conjunctive query answering in $\mathcal{EL}$ has not been studied yet, however the results in [14] show us that such a service is PTIME-hard in data complexity and hence cannot be delegated to a relational DBMS (actually such a lower bound holds already for instance checking).

Our work is also tightly related to work in databases on implication of integrity constraints (ICs) (see, e.g., [1, 22]) and on query answering in the presence of ICs under an open world semantics (see, e.g., [9, 3, 21, 8]). Rephrased as ICs, TBoxes in the *DL-Lite* family allow for expressing special forms of inclusion dependencies (i.e., ISA, role typing, and participation constraints), multiple keys on relations (i.e., functionality restrictions), and exclusion dependencies (i.e., disjointness and non-participation constraints)[11]. The results that we report here show that inclusion assertions allowed in the *DL-Lite* family form one of the largest class of ICs for which query answering remains polynomial.

One might ask whether significant new extensions to the DLs of the *DL-Lite* family are worth investigating. As shown in [14], one could add intersection to the left-hand side of concept inclusion assertions, without increasing the computational complexity of query answering. Also other extensions are interesting. The most relevant are discussed at the end of the next section. However, despite these cases, as also shown in [14], the DLs of the *DL-Lite* family are essentially the maximal DLs for which conjunctive query answering can be done in LOGSPACE, and that allow one to delegate query evaluation to a relational engine. In particular, the results reported in [14] imply that for the DL that includes all the constructs of the two logics studied in the paper, specifically, both inclusion assertions between roles and functionality assertions, query answering is PTIME-hard, thus ruling out the possibility of using off-the-shelf relational technology for query processing. In this sense, the *DL-Lite* family includes the first DLs specifically tailored for effective query answering over large amounts of data.

[11] This combination of ICs has only been studied in [8], but under a different semantics with respect to the one adopted in DLs.

8. Conclusions

We have presented the *DL-Lite* family, a new family of Description Logics specifically tailored to capture conceptual data models and basic ontology languages, while keeping the worst-case complexity of sound and complete reasoning tractable. We have concentrated our attention on two members of the family, namely, *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$. A notable feature of both logics is to allow for a separation between TBox and ABox reasoning. This separation enables an interesting modularization of query answering: the part of the process requiring TBox reasoning is independent of the ABox, and the part of the process requiring access to the ABox can be carried out by an SQL engine.

Based on this idea, we have developed a reasoning system for *DL-Lite*$_{\mathcal{F}}$, called QuOnto [2], whose main component is the query answering algorithm. In QuOnto, the information regarding the instances of a knowledge base are stored in a relational database. More precisely, given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we represent the interpretation $db(\mathcal{A})$ defined in Section 2 by means of the relational database $\mathsf{tab}_{db(\mathcal{A})}$, that, for each atomic concept A, stores $A^{db(\mathcal{A})}$ in a unary relation tab_A, and for each atomic role P, stores $P^{db(\mathcal{A})}$ in a binary relation tab_P. In this way, by eploiting the well-known correspondence between FOL queries and SQL queries, given a query q posed to $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the query computed by $\mathsf{PerfectRef}(q, \mathcal{T})$ is transformed into an equivalent SQL query to be executed over $\mathsf{tab}_{db(\mathcal{A})}$. This allows us to take advantage of the query optimizer that we find inside current DBMSs. Indeed, QuOnto has been tested within a research project carried out jointly by the University of Rome "La Sapienza" and the IBM Tivoli Laboratory, and the first experiments show that our approach is extremely effective: not only can we model complex domains in *DL-Lite*$_{\mathcal{F}}$, but it takes no more than a few minutes to answer complex conjunctive queries over knowledge bases with millions of instances.

As we have already said, the DLs of the *DL-Lite* family are essentially the maximal DLs for which conjunctive query answering can be done in LogSpace. Nevertheless, relatively interesting extensions to both *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$ exist that do not hamper the nice computational properties of the two logics. The most significant extensions are listed below.

1. The possibility of using conjunction in the concepts appearing in the left-hand side of inclusion assertions (we indicate the presence of such an extension by a subscript $\sqcap$ in the name of the logic).

2. The possibility of using n-ary relations, together with suitable constructs to deal with them, in addition to the binary roles dealt

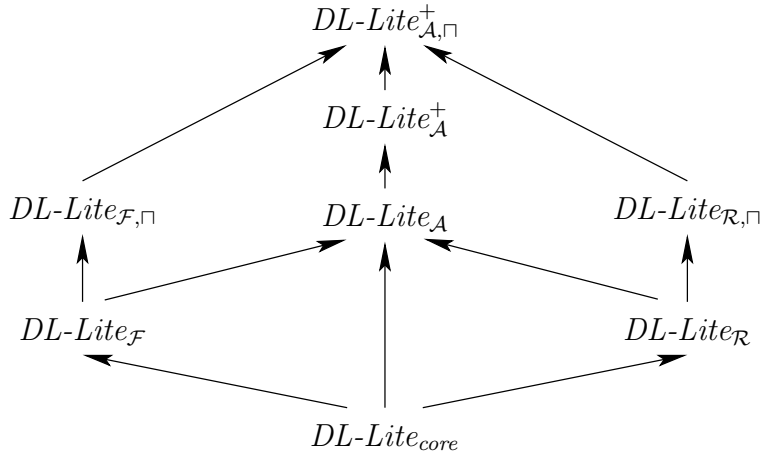

Figure 4. The *DL-Lite* family of description logics

with in this paper (we indicate the presence of such an extension by naming the logic $DLR\text{-}Lite_X$ instead of $DL\text{-}Lite_X$).

3. The possibility of using negative assertions and soft constants in unary predicates in the ABox. Note that these extensions pose new requirements on the method for storing the ABox using a DBMS; in particular, specific relations associated to negated concepts are needed in order to correctly represent negative assertions.

4. The possibility of using together role inclusion assertions and functionality assertions, with the limitation that functional roles are not specialized, i.e., do not occur in the right-hand side of role inclusion assertions.

5. The possibility of modeling attributes of concepts.

6. The possibility of modeling attributes of roles, as well as identification constraints.

The logics resulting from extension 1 and also the combination of extensions 1 and 2 are studied in [14], while extension 3 is considered in [18]. The logic combining extensions 4 and 5, called $DL\text{-}Lite_{\mathcal{A}}$, is studied in [28], while the logic combining, in addition to extensions 4 and 5, also extension 6, here called $DL\text{-}Lite^{+}_{\mathcal{A}}$, is studied in [12, 15].

The above extensions, and the corresponding logics provide a complete picture of the *DL-Lite* family. The logics of this family and their mutual relationships are depicted in Figure 4. A picture analogous to this figure can be obtained by replacing each logic $DL\text{-}Lite_X$ with its variant $DLR\text{-}Lite_X$ equipped with n-ary relations.

We point out that the technical development on *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$ presented in this paper constitutes the theoretical foundation for all the logics in this family, as well as for the QuOnto system. We are currently working on QuOnto in order to adapt it to the extensions mentioned above. Fortunately, the query answering algorithms illustrated in this paper, which are the critical components of the reasoning system, are not affected by the new features, and therefore remain valid.

Acknowledgments We thank the reviewers for many suggestions and comments on a previous version of the paper. This research has been partially supported by FET project TONES (Thinking ONtologiES), funded by the EU under contract number FP6-7603, by project HYPER, funded by IBM through a Shared University Research (SUR) Award grant, and by MIUR FIRB 2005 project "Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet" (TOCAI.IT).

References

1. Abiteboul, S., R. Hull, and V. Vianu: 1995, *Foundations of Databases*. Addison Wesley Publ. Co.
2. Acciarri, A., D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati: 2005, 'QuOnto: QUerying ONTOlogies'. In: *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*. pp. 1670–1671.
3. Arenas, M., L. E. Bertossi, and J. Chomicki: 1999, 'Consistent Query Answers in Inconsistent Databases'. In: *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*. pp. 68–79.
4. Baader, F., S. Brandt, and C. Lutz: 2005, 'Pushing the $\mathcal{EL}$ Envelope'. In: *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. pp. 364–369.
5. Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider (eds.): 2003, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.
6. Borgida, A. and R. J. Brachman: 2003, 'Conceptual Modeling with Description Logics'. in [5], Chapt. 10, pp. 349–372.
7. Borgida, A., R. J. Brachman, D. L. McGuinness, and L. A. Resnick: 1989, 'CLASSIC: A Structural Data Model for Objects'. In: *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*. pp. 59–67.
8. Bravo, L. and L. Bertossi: 2003, 'Logic Programming for Consistently Querying Data Integration Systems'. In: *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*. pp. 10–15.
9. Calì, A., D. Lembo, and R. Rosati: 2003a, 'On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases'. In: *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*. pp. 260–271.
10. Calì, A., D. Lembo, and R. Rosati: 2003b, 'Query Rewriting and Answering under Constraints in Data Integration Systems'. In: *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*. pp. 16–21.

11. Calvanese, D.: 1996, 'Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity'. In: *Proc. of the 12th Eur. Conf. on Artificial Intelligence (ECAI'96)*. pp. 303–307.
12. Calvanese, D., G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati: 2006a, 'Linking Data to Ontologies: The Description Logic DL-Lite$_A$'. In: *Proc. of the 2nd Workshop on OWL: Experiences and Directions (OWLED 2006)*.
13. Calvanese, D., G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati: 2005, 'DL-Lite: Tractable Description Logics for Ontologies'. In: *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*. pp. 602–607.
14. Calvanese, D., G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati: 2006b, 'Data Complexity of Query Answering in Description Logics'. In: *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*. pp. 260–270.
15. Calvanese, D., G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati: 2007, 'Can OWL Model Football Leagues?'. In: *Proc. of the 3rd Workshop on OWL: Experiences and Directions (OWLED 2007)*.
16. Calvanese, D., G. De Giacomo, and M. Lenzerini: 2000, 'Answering Queries Using Views over Description Logics Knowledge Bases'. In: *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*. pp. 386–391.
17. Chen, P. P.: 1976, 'The Entity-Relationship Model: Toward a Unified View of Data'. *ACM Trans. on Database Systems* **1**(1), 9–36.
18. De Giacomo, G., M. Lenzerini, A. Poggi, and R. Rosati: 2006, 'On the Update of Description Logic Ontologies at the Instance Level'. In: *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*.
19. Donini, F. M., M. Lenzerini, D. Nardi, and A. Schaerf: 1994, 'Deduction in Concept Languages: From Subsumption to Instance Checking'. *J. of Logic and Computation* **4**(4), 423–452.
20. Donini, F. M., M. Lenzerini, D. Nardi, and A. Schaerf: 1998, '$\mathcal{AL}$-log: Integrating Datalog and Description Logics'. *J. of Intelligent Information Systems* **10**(3), 227–252.
21. Duschka, O. M., M. R. Genesereth, and A. Y. Levy: 2000, 'Recursive Query Plans for Data Integration'. *J. of Logic Programming* **43**(1), 49–73.
22. Fan, W. and J. Siméon: 2000, 'Integrity Constraints for XML'. In: *Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2000)*. pp. 23–34.
23. Grosof, B. N., I. Horrocks, R. Volz, and S. Decker: 2003, 'Description Logic Programs: Combining Logic Programs with Description Logic'. In: *Proc. of the 12th Int. World Wide Web Conf. (WWW 2003)*. pp. 48–57.
24. Hustadt, U., B. Motik, and U. Sattler: 2005, 'Data Complexity of Reasoning in Very Expressive Description Logics'. In: *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. pp. 466–471.
25. Johnson, D. S. and A. C. Klug: 1984, 'Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies'. *J. of Computer and System Sciences* **28**(1), 167–189.
26. Levy, A. Y. and M.-C. Rousset: 1998, 'Combining Horn Rules and Description Logics in CARIN'. *Artificial Intelligence* **104**(1–2), 165–209.
27. Ortiz, M. M., D. Calvanese, and T. Eiter: 2006, 'Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics'. In: *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*.

28. Poggi, A., D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati: 2007, 'Linking Ontologies to Data'. *J. on Data Semantics*. To appear.
29. Rosati, R.: 2006, 'On the Decidability and Finite Controllability of Query Processing in Databases with Incomplete Information'. In: *Proc. of the 25th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2006)*. pp. 356–365.
30. Vardi, M. Y.: 1982, 'The Complexity of Relational Query Languages'. In: *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*. pp. 137–146.