Data Complexity of Query Answering in Description Logics*

Diego Calvanese[1], Giuseppe De Giacomo[2], Domenico Lembo[2], Maurizio Lenzerini[2], Riccardo Rosati[2]

[1] Faculty of Computer Science
Free University of Bozen-Bolzano
Piazza Domenicani 3, Bolzano, Italy
calvanese@inf.unibz.it

[2] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, Roma, Italy
lastname@dis.uniroma1.it

Abstract

In this paper we study data complexity of answering conjunctive queries over Description Logic knowledge bases constituted by an ABox and a TBox. In particular, we are interested in characterizing the FOL-reducibility and the polynomial tractability boundaries of conjunctive query answering, depending on the expressive power of the DL used to express the knowledge base. FOL-reducibility means that query answering can be reduced to evaluating queries over the database corresponding to the ABox. Since first-order queries can be expressed in SQL, the importance of FOL-reducibility is that, when query answering enjoys this property, we can take advantage of Data Base Management System (DBMS) techniques for both representing data, i.e., ABox assertions, and answering queries via reformulation into SQL. What emerges from our complexity analysis is that the Description Logics of the *DL-Lite* family are the maximal logics allowing conjunctive query answering through standard database technology. In this sense, they are the first Description Logics specifically tailored for effective query answering over very large ABoxes.

*This paper is an extended and abridged version of [10]

1 Introduction

The idea of using ontologies as a conceptual view over data repositories is becoming more and more popular. For example, in Enterprise Application Integration, Data Integration [20], and the Semantic Web [17], the intensional level of the application domain can be profitably represented by an ontology, so that clients can rely on a shared conceptualization when accessing the services provided by the system. In these contexts, the set of instances of the concepts in the ontology is to be managed in the data layer of the system architecture (e.g., in the lowest of the three tiers of the Enterprise Software Architecture), and, since instances correspond to the data items of the underlying information system, such a layer constitutes a very large (much larger than the intensional level of the ontology) repository, to be stored in secondary storage (see [9]).

When clients access the application ontology, it is very likely that one of the main services they need is the one of answering complex queries over the extensional level of the ontology (obviously making use of the intensional level as well in producing the answer). Here, by complex we mean that it does not suffice to ask for the instances of concepts, but we need at least expressing conjunctive conditions on the extensional level. Given the size of the instance repository, when measuring the computational complexity of query answering (and reasoning in general) the most important parameter is the size of the data. In other words, we are interested in the so-called *data complexity* of query answering.

In this paper we consider conjunctive queries (CQs) specified over ontologies expressed in Description Logics (DL), and study the data complexity of the query answering problem. Since an ontology in DL is essentially a knowledge base (KB) constituted by a TBox and an ABox, the problem we address is the one of computing the answers to a CQ that are logical consequences of the TBox and the ABox, where complexity is measured with respect to the size of the ABox only. Note that we borrow the notion of data complexity from the database literature [23], on the premise that an ABox can be naturally viewed as a relational database.

We are interested in characterizing the FOL-reducibility and the polynomial tractability boundaries of conjunctive query answering, depending on the expressive power of the DL used to express the KB. We say that query answering is FOL-reducible in a DL $\mathcal{L}$, if for every conjunctive query q over an $\mathcal{L}$ TBox $\mathcal{T}$, there is a first-order query q' such that for all ABoxes $\mathcal{A}$ the answers to q with respect to the KB $(\mathcal{T}, \mathcal{A})$ are the same as the answers to q' over the database corresponding to the ABox $\mathcal{A}$. Since first-order queries can be expressed in SQL, the importance of FOL-reducibility is that, when query answering enjoys this property, we can take advantage of Data Base Management System (DBMS) techniques for both representing data, i.e., ABox assertions, and an-

swering queries via reformulation into SQL[1]. Notably, in this case, the data complexity of conjunctive query answering over ontologies is the one of FOL queries over databases, i.e., LogSpace.

We are also interested to know for which DLs we go beyond FOL. For this purpose, we consider the LogSpace boundary of the problem. Indeed, we single out those DLs for which query answering becomes NLogSpace-hard and PTime,-hard respectively. From the complexity characterization of query languages, it follows that those DLs require at least the power of linear recursive Datalog (NLogSpace), and general recursive Datalog (PTime). Note that, although very interesting and promising Datalog engines exist, query optimization strategies for this query language are not sufficiently mature yet to deal with complex applications with millions of instances in the extensional level. Finally, we address the problem of going even beyond PTime, by exhibiting DLs for which query answering is polynomially intractable.

More precisely, the contributions of the paper are the following.

- We discuss DLs for which conjunctive query answering is FOL-reducible. In this class, we essentially find the family of *DL-Lite* [11] languages. This family is constituted by two simple DLs, which are rich enough to express basic ontology languages, e.g., extensions of (the DL subset of) RDFS [6] or fragments of OWL-DL [5]; conceptual data models, e.g., Entity-Relationship [7]; and object-oriented formalisms, e.g., basic UML class diagrams [3]. We also show that we can extend these two languages by adding n-ary relations, and still retain FOL-reducibility. We show that the DLs of the *DL-Lite* family are maximally expressive DLs for which query answering is FOL reducible.

- We show that minimal additions to the languages considered above bring data complexity of conjunctive query answering to NLogSpace-hardness and PTime-hardness, thus losing the possibility of reformulating queries in first-order logic. In spite of the fact that we conjecture that for such languages query answering is polynomially tractable (in NLogSpace and PTime, respectively), these hardness results tell us that in query answering we cannot take advantage of state-of-the-art database query optimization strategies, and this might hamper practical feasibility for very large ABoxes.

- Finally, we establish coNP-hardness of conjunctive query answering with respect to data complexity for surprisingly simple DLs. In particular, we show that we get intractability as soon as the DL is able to express simple forms of union.

[1] We consider here the kernel of the SQL-92 standard, i.e., we see SQL as an implementation of relational algebra.

What emerges from our complexity analysis is that the two versions of *DL-Lite* are the maximal DLs which allow for answering conjunctive queries through standard database technology. In this sense, they are the first DLs specifically tailored for effective query answering over large amounts of data.

The paper is organized as follows. In the next section we introduce some preliminaries which will be useful for the subsequent discussions. In Section 3 and 4, we present DLs for which query answering is FOL-reducible. Then, we deal with DLs for which query answering goes beyond LogSpace: in Section 5 we identify DLs for which query answering is NLogSpace-hard; in Section 6 we characterize DLs for which query answering is PTime-hard; and in Section 7 we identify DLs for which query answering is coNP-hard. In Section 8 we overview related work, and in Section 9 we draw some conclusions.

2 Preliminaries

Description Logics (DLs) [8] 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. In this paper, we distinguish between the constructs that are allowed in the concepts in the left-hand side (Cl) and those in the right-hand side (Cr) of inclusion assertions (see later).

As a concrete example of a DL, we focus on *DL-Lite*$_{core}$, which serves as core language for the family of *DL-Lite* languages discussed in the rest of the paper. The language for *DL-Lite*$_{core}$ concepts and roles is defined as follows:

$$\begin{array}{rcl} Cl & \longrightarrow & A \mid \exists R \mid Cl_1 \sqcap Cl_2 \mid Cl_1 \sqcup Cl_2 \mid \bot \\ Cr & \longrightarrow & A \mid \exists R \mid Cr_1 \sqcap Cr_2 \mid \bot \mid \top \\ R & \longrightarrow & P \mid P^- \end{array}$$

where Cl (resp., Cr) denotes a concept used in the left-hand side (resp., right-hand side) of an inclusion assertion, A denotes an atomic concept, P an atomic role, and P^- its inverse[2].

The semantics of a DL, e.g., *DL-Lite*$_{core}$, is given in terms of interpretations, where an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of an interpretation domain $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of

[2]We observe that including $Cl_1 \sqcup Cl_2$ in the constructs for the left-hand side of the inclusion assertions and $Cr_1 \sqcap Cr_2$ in the constructs for the right-hand side does not extend the expressive capabilities of the language, since these can be simulated by considering that $Cl_1 \sqcup Cl_2 \sqsubseteq Cr$ is equivalent to the pair of assertions $Cl_1 \sqsubseteq Cr$ and $Cl_2 \sqsubseteq Cr$, and that $Cl \sqsubseteq Cr_1 \sqcap Cr_2$ is equivalent to $Cl \sqsubseteq Cr_1$ and $Cl \sqsubseteq Cr_2$. Similarly, we can drop $\bot$ from the constructs for the left-hand side and $\top$ from those for the right-hand side.

$\Delta^{\mathcal{I}}$, and to each role R a binary relation over $\Delta^{\mathcal{I}}$. In particular for the constructs of *DL-Lite*$_{core}$ we have:

$$\begin{array}{lcllcl} \top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} & (C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\ \bot^{\mathcal{I}} & = & \emptyset & (C_1 \sqcup C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\ A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} & (\exists R)^{\mathcal{I}} & = & \{o \mid \exists o' . (o, o') \in R^{\mathcal{I}}\} \\ P^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} & (P^-)^{\mathcal{I}} & = & \{(o_2, o_1) \mid (o_1, o_2) \in P^{\mathcal{I}}\} \end{array}$$

A DL *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ represents the domain of interest and consists of two parts, a *TBox* $\mathcal{T}$, representing intensional knowledge, and an *ABox* $\mathcal{A}$, representing extensional knowledge. A TBox is formed by a set of *inclusion assertions* of the form

$$Cl \sqsubseteq Cr$$

where Cl and Cr are formed using the constructs allowed by the particular DL used, e.g., for *DL-Lite*$_{core}$ we can have the constructs described above. Such an inclusion assertion expresses that all instances of concept Cl are also instances of concept Cr. Apart from the above inclusion assertions, some DLs that we consider in this paper allow for other forms of assertions in the TBox (see later). *DL-Lite*$_{core}$ does not allow for any other form of assertion however.

An ABox is formed by a set of *membership assertions* on atomic concepts and on atomic roles:

$$A(a), \qquad P(a_1, a_2)$$

stating respectively that the object (denoted by the constant) a is an instance of A and that the pair (a_1, a_2) of objects is an instance of the role P. Other forms of ABoxes have also been proposed [8], but we will not consider them here.

Formally, an interpretation $\mathcal{I}$ is a *model* of an inclusion assertion $Cl \sqsubseteq Cr$ if $Cl^{\mathcal{I}} \subseteq Cr^{\mathcal{I}}$. We extend the interpretation function to constants, by assigning to each constant a a *distinct* object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$; in other words, as usual in DLs, we enforce the *unique name assumption* on constants [8]. An interpretation $\mathcal{I}$ is a model of a membership assertion $A(a)$ (resp., $P(a_1, a_2)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp., $(a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in P^{\mathcal{I}}$). A *model of a KB* $\mathcal{K}$ is an interpretation $\mathcal{I}$ that is a model of all assertions in $\mathcal{K}$. 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.

Given a KB $\mathcal{K}$ expressed in a DL, we can query it by using queries. In particular we will concentrate on conjunctive queries: A *conjunctive query* $q(\vec{x})$ over a KB $\mathcal{K}$ is an expression of the form

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

where $\vec{x}$ are the so-called distinguished variables (which will be bound with objects in the KB), $\vec{y}$ are the non-distinguished variables (which are existentially

quantified), and $conj(\vec{x}, \vec{y})$ is a conjunction of atoms of the form $A(z)$ or $P(z_1, z_2)$ where A and P are respectively atomic concepts and roles of $\mathcal{K}$ and z, z_1, z_2 are either constants in $\mathcal{K}$ or variables in $\vec{x}$ or $\vec{y}$.

Given an interpretation $\mathcal{I}$, the conjunctive query $q(\vec{x}) = \{\vec{x} \mid conj(\vec{x}, \vec{y})\}$ is interpreted as the set $q^{\mathcal{I}}$ of tuples $\vec{o}$ of objects such that, when assigning $\vec{o}$ to $\vec{x}$, the first-order formula $\exists \vec{y}.conj(\vec{x}, \vec{y})$ evaluates to true in $\mathcal{I}$.

The reasoning service we are interested in is *(conjunctive) query answering*: given a knowledge base $\mathcal{K}$ and a conjunctive query $q(\vec{x})$ over $\mathcal{K}$, return all tuples $\vec{a}$ of constants in $\mathcal{K}$ such that, when substituted to the variables $\vec{x}$ in $q(\vec{x})$, we have that $\mathcal{K} \models q(\vec{a})$, i.e., such that $\vec{a}^{\mathcal{I}} \in q^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. We observe that query answering (properly) generalizes a well known reasoning service in DLs, namely *instance checking*, i.e., logical implication of an ABox assertion. In particular, instance checking can be expressed as the problem of answering (boolean) conjunctive queries constituted by just one ground atom.

Finally, we refer to *data complexity* of query answering, which is a notion borrowed from relational database theory [23]. First, we note that there is a recognition problem associated with query answering, which is defined as follows. We have a fixed TBox $\mathcal{T}$ expressed in a DL $\mathcal{L}$, and a fixed query q: the *recognition problem* associated to $\mathcal{T}$ and q is the decision problem of checking whether, given an ABox $\mathcal{A}$, and a tuple $\vec{a}$ of constants, we have that $(\mathcal{T}, \mathcal{A}) \models q(\vec{a})$. Note that neither the TBox nor the query is an input to the recognition problem.

Let $\mathcal{S}$ be a complexity class. When we say that query answering for a certain DL $\mathcal{L}$ *is in* $\mathcal{S}$ *with respect to data complexity*, we mean that the corresponding recognition problem is in $\mathcal{S}$. Similarly, when we say that query answering for a certain DL $\mathcal{L}$ is $\mathcal{S}$*-hard with respect to data complexity*, we mean that the corresponding recognition problem is $\mathcal{S}$-hard.

We will also use the notion of Q*-reducibility* of query answering, where Q is a given query language. Query answering in a DL $\mathcal{L}$ is Q*-reducible* if for every (conjunctive) query q and every TBox $\mathcal{T}$ expressed in $\mathcal{L}$, there exists a query q_1, over the same alphabet, belonging to the query language Q, such that for every ABox $\mathcal{A}$ $(\mathcal{T}, \mathcal{A}) \models q(\vec{a})$ iff $\vec{a}^{\mathcal{I}_\mathcal{A}} \in q_1^{\mathcal{I}_\mathcal{A}}$, where $\mathcal{I}_\mathcal{A}$ is an interpretation defined as follows: $a^{\mathcal{I}_\mathcal{A}} = a$ for each constant a, $A^{\mathcal{I}_\mathcal{A}} = \{a \mid A(a) \in \mathcal{A}\}$ for each atomic concept A, and $P^{\mathcal{I}_\mathcal{A}} = \{(a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A}\}$ for each atomic role P. In other words, q_1 is evaluated over the ABox $\mathcal{A}$ considered as a database. One of the most interesting classes of queries is that of FOL queries, i.e., the queries expressed in first-order logic, since, from the practical point of view, FOL queries correspond to queries expressed in relational algebra (i.e., in SQL). Observe that every FOL query can be evaluated in LogSpace wrt data complexity (see e.g., [7]). It follows that if $\mathcal{L}$ is FOL-reducible then query answering in $\mathcal{L}$ is in LogSpace wrt data complexity. Vice-versa, if query answering is $\mathcal{S}$-hard wrt data complexity for some complexity class $\mathcal{S}$ larger than LogSpace (e.g., NLogSpace, PTime, coNP, etc.) then it is not FOL-reducible.

3 FOL-reducibility of *DL-Lite*

In this section we discuss two new DLs that extend *DL-Lite*$_{core}$, and show that in such DLs query answering is FOL-reducible (and hence is in LogSpace).

The first DL which we consider is *DL-Lite*$_{\mathcal{F}}$, which has the same language (i.e., the same concept and role constructs) as *DL-Lite*$_{core}$, but additionally allows for assertions of the form (*funct* R) in the TBox, expressing functionality of role R. An interpretation $\mathcal{I}$ is a model of an assertion (*funct* P) if the binary relation $P^{\mathcal{I}}$ is a function, i.e., $(o, o_1) \in P^{\mathcal{I}}$ and $(o, o_2) \in P^{\mathcal{I}}$ implies $o_1 = o_2$. Analogously for (*funct* P^-).

Notice that *DL-Lite*$_{\mathcal{F}}$ is actually an extension of the DL presented in [11] (simply called *DL-Lite*), which essentially did not have conjunctions in $\mathcal{C}l$. In [11] we have presented an algorithm for query answering based on the idea of expanding the original query into a set (i.e., a union) of conjunctive queries that can be directly evaluated over the ABox. The expansion process takes into account only the original query and the TBox assertions, and is independent of the ABox, which can be easily managed in secondary storage by a relational DBMS. Therefore, from the results in [11] it follows that query answering in *DL-Lite* is FOL-reducible. In the following, we show that FOL-reducibility of query answering still holds in *DL-Lite*$_{\mathcal{F}}$, as stated by the theorem below.

Theorem 1 *Query answering in DL-Lite*$_{\mathcal{F}}$ *is FOL-reducible and therefore is in* LogSpace *with respect to data complexity.*

Proof. The proof is based on an algorithm for conjunctive query answering over *DL-Lite*$_{\mathcal{F}}$ knowledge bases, which relies on a technique for rewriting the input query in FOL, which is obtained by extending the reformulation technique of *DL-Lite* presented in [11].

Such an extension is analogous to the one described in the proof of Theorem 2, which is given in the following. Indeed, a part for functionalities on roles, a *DL-Lite*$_{\mathcal{F}}$ knowledge base is actually a special case of a *DL-Lite*$_{\mathcal{R}}$ knowledge base (in which neither inclusion assertions between roles nor qualified existential quantification on the right-hand side of inclusions between concepts are allowed). As for functionalities on roles, we point out that Lemma 5 introduced in the proof of Theorem 2 also holds for *DL-Lite*$_{\mathcal{F}}$ knowledge bases. In this language however, the TBox $\mathcal{T}_{NI}$ also contains functionality assertions (whereas $\mathcal{T}_{PI}$ is obtained from $\mathsf{Normalize}(\mathcal{T})$ by dropping both negative inclusions and functionality assertions). As a consequence, we can still apply the algorithms Consistent and PerfectRef, provided that Consistent also computes a FOL query for each functionality assertion (together with a FOL query for each NI as for *DL-Lite*$_{\mathcal{R}}$ knowledge bases). More precisely, given an input query q of arity n, for each functionality assertion (*funct* P) belonging to $\mathcal{T}_{NI}$, Consistent computes

the FOL query

$$q_\phi(x_1, \ldots, x_n) = \{x_1, \ldots, x_n \mid \exists y, z.P(x, y) \wedge P(x, z) \wedge y \neq z \wedge val(x_1) \wedge \ldots \wedge val(x_n)\}.$$

Analogously for assertions of the form (*funct* P^-). □

Notice that *DL-Lite*$_{core}$ only allows for unqualified existential quantification and inclusions between concepts. One might ask what happens to query answering if we add qualified existential quantification and inclusions between roles to the language. To this purpose, we consider a second notable extension of *DL-Lite*$_{core}$, called *DL-Lite*$_{\mathcal{R}}$. The DL *DL-Lite*$_{\mathcal{R}}$ has the same language of *DL-Lite*$_{core}$ for the roles and for the concepts on the left-hand side of inclusion assertions, while concepts on the right-hand side are formed according to the following syntax:

$$Cr \longrightarrow A \mid \exists R.Cr \mid Cr_1 \sqcap Cr_2 \mid \bot \mid \top$$

For each interpretation $\mathcal{I}$, besides those of *DL-Lite*$_{core}$, the equation $(\exists R.Cr)^{\mathcal{I}} = \{o \mid \exists o'.\, (o, o') \in R^{\mathcal{I}} \text{ and } o' \in Cr^{\mathcal{I}}\}$ holds.

In addition to inclusion assertions between concepts, *DL-Lite*$_{\mathcal{R}}$ allows for inclusion assertions between roles of the form:

$$R_1 \sqsubseteq R_2$$

where R_i is either an atomic role or its inverse. An interpretation $\mathcal{I}$ is a model of such an assertion if $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$.

For the above DL the following result holds.

Theorem 2 *Query answering in DL-Lite*$_{\mathcal{R}}$ *is FOL-reducible and therefore is in* LogSpace *with respect to data complexity.*

Proof. To prove the thesis we provide an algorithm which, taken as input a *DL-Lite*$_{\mathcal{R}}$ TBox $\mathcal{T}$ and a conjunctive query q specified over $\mathcal{T}$, returns a union of conjunctive queries (and therefore a FOL query) q_1 such that, for each ABox $\mathcal{A}$, the evaluation of q_1 over the ABox $\mathcal{A}$ considered as a database (see Section 2) returns the set of tuples in the answer to q over the knowledge base $(\mathcal{T}, \mathcal{A})$, i.e., returns all tuples $\vec{a}$ such that $(\mathcal{T}, \mathcal{A}) \models q(\vec{a})$. We prove soundness and completeness of the algorithm with respect to the problem of computing such set of tuples.

As usual, with *Cl* (resp. *Cr*) we denote a concept used in the left-hand side (resp. right-hand side) of inclusion assertions between concepts, with A we denote an atomic concept, with P an atomic role, and with P^- its inverse (whereas R indicates either P or P^-). Furthermore, we use the symbol B to denote *basic concepts*, i.e., we use B to indicate either A, $\exists P$, or $\exists P^-$. Also,

without loss of generality, we assume that, in what follows, every concept name or role name occurring in an ABox $\mathcal{A}$ also occurs in the corresponding TBox $\mathcal{T}$. Finally, we recall that a FOL query $q(\vec{x})$ over a KB $\mathcal{K}$ is an expression of the form

$$\{\, \vec{x} \;\mid\; body_q(\vec{x}, \vec{y}) \,\}$$

where $\vec{x}$ are the so-called distinguished variables (which will be bound with objects in the KB), $\vec{y}$ are the non-distinguished variables (which are existentially quantified), and $body_q(\vec{x}, \vec{y})$ is a FOL formula involving atoms of the form $A(z)$ or $P(z_1, z_2)$ where A and P are respectively atomic concepts and roles of $\mathcal{K}$ and z, z_1, z_2 are either constants in $\mathcal{K}$ or variables in $\vec{x}$ or $\vec{y}$[3].

The algorithm makes use of three main functions, namely Normalize, Consistent, and PerfectRef, which correspond to three phases called *Normalization*, *Satisfiability check* and *Query reformulation*, respectively. The first one performs some preliminary transformations on the TBox $\mathcal{T}$. The second one computes a portion of the final output query that properly deals with situations in which the ABox $\mathcal{A}$, over which the final output query is evaluated, contradicts the TBox $\mathcal{T}$, i.e., the knowledge base $(\mathcal{T}, \mathcal{A})$ is unsatisfiable. Notice that, in these cases, every n-tuple of constants of $\mathcal{A}$ is in the answer to every query of arity n over $\mathcal{T}$. Finally, the third function computes the remaining portion of the output query. Roughly speaking, PerfectRef reformulates the input query q into a FOL query in which it compiles the knowledge of the TBox $\mathcal{T}$ that is needed to answer q.

Normalization. The function Normalize takes as input the TBox $\mathcal{T}$ and transforms it as follows:

1. replaces each assertion of the form $Cl_1 \sqcup Cl_2 \sqsubseteq Cr$ with the assertions
$$\begin{array}{lll} Cl_1 & \sqsubseteq & Cr \\ Cl_2 & \sqsubseteq & Cr; \end{array}$$
2. replaces each assertion of the form $Cl \sqsubseteq Cr_1 \sqcap Cr_2$ with the assertions
$$\begin{array}{lll} Cl & \sqsubseteq & Cr_1 \\ Cl & \sqsubseteq & Cr_2; \end{array}$$
3. erases each assertion of the form $\bot \sqsubseteq Cr$, $\bot \sqcap Cl \sqsubseteq Cr$, or $Cl \sqsubseteq \top$;
4. replaces each assertion of the form $Cl \sqsubseteq \exists P.Cr$ (resp. $Cl \sqsubseteq \exists P^-.Cr$) with the assertions
$$\begin{array}{ll} Q \sqsubseteq P & \\ \exists Q^- \sqsubseteq Cr & (\text{resp. } \exists Q \sqsubseteq Cr) \\ Cl \sqsubseteq \exists Q & (\text{resp. } Cl \sqsubseteq \exists Q^-); \end{array}$$

[3]Obviously, for FOL queries that are conjunctive queries, $body_q(\vec{x}, \vec{y})$ is a conjunction of atoms, that can be also denoted with $\exists y.conj(\vec{x}, \vec{y})$ (see Section 2).

5. closes the TBox with respect to the following inference rules:

   (*i*) if $Cl \sqsubseteq B$ and $B \sqcap Cl' \sqsubseteq \bot$ occur in $\mathcal{T}$, then add $Cl \sqcap Cl' \sqsubseteq \bot$ to $\mathcal{T}$;

   (*ii*) if $P_1 \sqsubseteq P_2$ or $P_1^- \sqsubseteq P_2^-$ and $\exists P_2 \sqcap Cl \sqsubseteq \bot$ (resp. $\exists P_2^- \sqcap Cl \sqsubseteq \bot$) belong to $\mathcal{T}$, then add $\exists P_1 \sqcap Cl \sqsubseteq \bot$ (resp. $\exists P_1^- \sqcap Cl \sqsubseteq \bot$) to $\mathcal{T}$, or if $P_1^- \sqsubseteq P_2$ or $P_1 \sqsubseteq P_2^-$ and $\exists P_2 \sqcap Cl \sqsubseteq \bot$ (resp. $\exists P_2^- \sqcap Cl \sqsubseteq \bot$) belong to $\mathcal{T}$, then add $\exists P_1^- \sqcap Cl \sqsubseteq \bot$ (resp. $\exists P_1 \sqcap Cl \sqsubseteq \bot$) to $\mathcal{T}$.

In the following, we denote with $\mathsf{Normalize}(\mathcal{T})$ the TBox obtained after processing $\mathcal{T}$ according to the above steps.

We are now able to prove some notable properties that hold for a normalized TBox. In particular, we show that $\mathcal{T}$ and $\mathsf{Normalize}(\mathcal{T})$ are "equivalent" with respect to conjunctive query answering, as formally stated below.

Lemma 3 *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox, q a conjunctive query over $\mathcal{T}$, $\mathcal{A}$ an ABox, and $\vec{a}$ a tuple of constants occurring in $\mathcal{A}$. Then, $\vec{a}$ is in the answer to q over the knowledge base $(\mathcal{T}, \mathcal{A})$ iff $\vec{a}$ is in the answer to q over the knowledge base $(\mathsf{Normalize}(\mathcal{T}), \mathcal{A})$, i.e., $(\mathcal{T}, \mathcal{A}) \models q(\vec{a})$ iff $(\mathsf{Normalize}(\mathcal{T}), \mathcal{A}) \models q(\vec{a})$.*

Notice that, according to steps from 1 to 4 of the algorithm, the TBox $\mathsf{Normalize}(\mathcal{T})$ contains only assertions of the form (*i*) $Cl \sqsubseteq B$, (*ii*) $R_1 \sqsubseteq R_2$, and (*ii*) $Cl \sqsubseteq \bot$. We call *positive inclusions (PIs)* inclusions of the forms (*i*) and (*ii*), and *negative inclusions (NIs)* inclusions of form (*iii*). The aim of step 5 is to expand the TBox by computing all (non-trivial) NIs logically implied by $\mathcal{T}$. We now prove that that the inference rules given at point 6 are sound and complete with respect to logical implication of NIs.

Lemma 4 *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox. Then, for every sequence $B_1 \sqcap \ldots \sqcap B_n$ of basic concepts, $\mathcal{T} \models (\mathcal{B}_1 \sqcap \ldots \sqcap B_n \sqsubseteq \bot)$ iff $(\mathcal{B}_1 \sqcap \ldots \sqcap B_n \sqsubseteq \bot) \in \mathsf{Normalize}(\mathcal{T})$.*

In the following, we indicate with $\mathcal{T}_{PI}$ the TBox obtained by dropping all negative inclusions from $\mathsf{Normalize}(\mathcal{T})$, and with $\mathcal{T}_{NI}$ the TBox obtained by dropping all positive inclusions from $\mathsf{Normalize}(\mathcal{T})$. It is easy to see that $\mathcal{T}_{NI}$ and $\mathcal{T}_{PI}$ are disjoint and that $\mathsf{Normalize}(\mathcal{T}) = \mathcal{T}_{PI} \cup \mathcal{T}_{NI}$.

Finally, we prove below that, in order to answer conjunctive queries over a *DL-Lite$_\mathcal{R}$* knowledge base, we can separately "reason" on NIs and PIs.

Lemma 5 (Separation) *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox, q a conjunctive query over $\mathcal{T}$, $\mathcal{A}$ an ABox, and $\vec{a}$ a tuple of constants occurring in $\mathcal{A}$. Then, $(\mathcal{T}, \mathcal{A}) \not\models q(\vec{a})$ iff $(\mathcal{T}_{NI}, \mathcal{A})$ is satisfiable and $(\mathcal{T}_{PI}, \mathcal{A}) \not\models q(\vec{a})$.*

We point out that $(\mathcal{T}, \mathcal{A})$ is unsatisfiable if and only if $(\mathcal{T}_{NI}, \mathcal{A})$ is unsatisfiable. Therefore, checking satisfiability of $(\mathcal{T}_{NI}, \mathcal{A})$ means actually checking satisfiability of $(\mathcal{T}, \mathcal{A})$.

According to the above theorem, to answer a conjunctive query q expressed over a *DL-Lite*$_{\mathcal{R}}$ knowledge base $(\mathcal{T}, \mathcal{A})$ we can proceed in two steps: (i) checking satisfiability of $(\mathcal{T}_{NI}, \mathcal{A})$ and then (ii) answering q as NIs were not specified over the TBox $\mathcal{T}$. Since our aim is to show FOL-reducibility of query answering for *DL-Lite*$_{\mathcal{R}}$ knowledge bases, we provide in the following two functions that allow for achieving the above goals by means of suitable FOL queries (that "put together" constitute the FOL reduction of the input query q).

Satisfiability check. The function Consistent is in charge of properly dealing with situations in which the ABox $\mathcal{A}$ contradicts (at least one) NIs of the TBox $\mathcal{T}$, i.e., $(\mathcal{T}_{NI}, \mathcal{A})$ is unsatisfiable. Observe that in such a case query answering is meaningless, since, according to the "ex falso quod libet" principle, every tuple is in the answer to every query (of the same arity). Therefore, with regards to this issue, the function Consistent takes as input the TBox $\mathcal{T}_{NI}$ and a query q of arity n and proceeds as follows[4]:

1. for each NI inclusion $\iota = B_1 \sqcap \ldots \sqcap B_m \sqsubseteq \bot$ belonging to $\mathcal{T}_{NI}$, it computes the FOL query

   $$q_\iota(x_1, \ldots, x_n) = \{x_1, \ldots, x_n \mid \exists y. C_1(y) \wedge \ldots \wedge C_m(y) \wedge val(x_1) \wedge \ldots \wedge val(x_n)\}$$

   where

   - for each $i \in \{1, \ldots, m\}$, $C_i(y) = A_i(y)$ if $B_i = A_i$, or $C_i(y) = \exists z_i. P_i(y, z_i)$ if $B_i = \exists P_i$ or $C_i(y) = \exists z_i. P_i(z_i, y)$ if $B_i = \exists P_i^-$, and
   - for each $i \in \{1, \ldots, n\}$, $val(x_i) = A_1(x_i) \vee \ldots \vee A_\ell(x_i) \vee \exists w_1. R_1(x_i, w_1) \vee \ldots \vee \exists w_k. R_k(x_i, w_k) \vee \exists v_1. R_1(v_1, x_i) \vee \ldots \vee \exists v_k. R_k(v_k, x_i)$, where $A_1, \ldots, A_\ell$ and $R_1, \ldots, R_k$ are all the atomic concepts and the atomic roles over which inclusions of the TBox $\mathcal{T}$ are asserted;

2. returns the query

   $$q_c(\vec{x}) = \{\vec{x} \mid body_{q_\alpha}(\vec{x}, \vec{y_\alpha}) \vee \ldots \vee body_{q_\nu}(\vec{x}, \vec{y_\nu})\}$$

   where with $body_{q_\alpha}(\vec{x}, \vec{y_\alpha}), \ldots, body_{q_\nu}(\vec{x}, \vec{y_\nu})$ we denote the bodies of queries $q_\alpha, \ldots, q_\nu$ constructed according to step 1.

Remember (see Section 2) that, in order to answer the query q over a *DL-Lite*$_{\mathcal{R}}$ knowledge base $(\mathcal{T}, \mathcal{A})$, the above query will be evaluated over the ABox

[4]Actually, Consistent only makes use of the arity n of q.

$\mathcal{A}$ considered as a database, i.e., over the interpretation $\mathcal{I}_\mathcal{A}$ defined as follows: $a^{\mathcal{I}_\mathcal{A}} = a$ for each constant a occurring in $\mathcal{A}$, $A^{\mathcal{I}_\mathcal{A}} = \{a \mid A(a) \in \mathcal{A}\}$ for each atomic concept A, and $P^{\mathcal{I}_\mathcal{A}} = \{(a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A}\}$ for each atomic role P. Given a knowledge base $(\mathcal{T}, \mathcal{A})$ we will call the interpretation $\mathcal{I}_\mathcal{A}$ defined above, the *minimal interpretation of* $\mathcal{A}$. Then, it should be easy to see that, if $\mathcal{A}$ contradicts a NI $\iota = B_1 \sqcap \ldots \sqcap B_m \sqsubseteq \bot$ of the TBox $\mathcal{T}$ (and therefore contained in $\mathcal{T}_{NI}$), there exists a substitution for y in the body of q_ι with a constant d occurring in $\mathcal{A}$ such that $C_1(d) \wedge \ldots \wedge C_m(d)$ evaluates to true in $\mathcal{I}_\mathcal{A}$. Then, any n-tuples constructible from constants occurring in $\mathcal{A}$ is returned by the evaluation of q_ι over $\mathcal{I}_\mathcal{A}$, due to the conjunction of predicates $val(x_1) \wedge \ldots \wedge val(x_n)$ in the body of q_ι. Then, the following Lemma is easily proved.

Lemma 6 *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox, q a conjunctive query of arity n over $\mathcal{T}$, and let $\mathcal{A}$ be an ABox. Then, $(\mathcal{T}_{NI}, \mathcal{A})$ is unsatisfiable iff* Consistent$(\mathcal{T}, q)$ *returns a query q_c such that $q_c^{\mathcal{I}_\mathcal{A}}$ returns all n-tuples of constants constructible from the constants of $\mathcal{A}$.*

Since Consistent provides the right answers to the query for all cases in which the ABox contradicts NIs of the TBox, in the following we can focus our attention to the different case in which the knowledge base is satisfiable (and therefore we will consider only ABoxes $\mathcal{A}$ such that $(\mathcal{T}_{NI}, \mathcal{A})$ is satisfiable).

Query reformulation. Query reformulation is achieved by means of the algorithm PerfectRef. The basic idea of our method is to reformulate the input query expressed over the TBox $\mathcal{T}$ taking into account only the PIs in $\mathcal{T}$. In particular, given a query q over a *DL-Lite*$_\mathcal{R}$ knowledge base $(\mathcal{T}, \mathcal{A})$, we compile the PIs of $\mathcal{T}$ into the query itself, thus obtaining a new query q_r. The evaluation of such a new query q_r over the ABox $\mathcal{A}$ considered as a simple relational database, i.e., over the minimal interpretation $\mathcal{I}_\mathcal{A}$ of $\mathcal{A}$, returns the answer to q over $(\mathcal{T}, \mathcal{A})$ (when $(\mathcal{T}, \mathcal{A})$ is satisfiable).

In the following, we illustrate PerfectRef from a technical point of view.

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, while we say that it is *unbound* if it corresponds to a non-distinguished non-shared variable (we use the symbol $\_$ to represent non-distinguished non-shared variables).

We now specify when a positive inclusion *is applicable to a query atom g*: (i) a PI I is applicable to an atom of the form $A(x)$, where A is an atomic concept, if I is of the form $Cl \sqsubseteq A$; (ii) a PI I is applicable to an atom $P_1(x_1, x_2)$, if one of the three following conditions holds (a) $x_2 = \_$ and I is of the form $Cl \sqsubseteq \exists P_1$, (b) $x_1 = \_$ and I is of the form $Cl \sqsubseteq \exists P_1^-$, (c) I is of the form $R_2 \sqsubseteq P_1$ or $R_2 \sqsubseteq P_1^-$, where R_2 can be either P_2 or P_2^-. Roughly speaking, an inclusion I is applicable to an atom g if all bound arguments of g are propagated by I.

We indicate with $gr(g, I)$ the atom obtained from the atom g by applying the inclusion I, i.e.,

- if $I = B_1 \sqcap \ldots \sqcap B_m \sqsubseteq A$ (resp., $I = B_1 \sqcap \ldots \sqcap B_m \sqsubseteq \exists P_1$ or $I = B_1 \sqcap \ldots \sqcap B_m \sqsubseteq \exists P_1^-$), where each B_i is a basic concept, and if $g = A(x)$ (resp., $g = P_1(x,\_)$ or $g = P_1(\_, x)$), we have $gr(g, I) = C_1(x) \wedge \ldots \wedge C_m(x)$, where for each $i \in \{1, \ldots, m\}$, $C_i(x) = A_i(x)$ if $B_i = A_i$, or $C_i(y) = \exists z_i.P_i(y, z_i)$ if $B_i = \exists P_i$ or $C_i(y) = \exists z_i.P_i(z_i, y)$ if $B_i = \exists P_i^-$;

- if $I = P_1 \sqsubseteq P_2$ or $I = P_1^- \sqsubseteq P_2^-$ (resp. $I = P_1^- \sqsubseteq P_2$ or $I = P_1 \sqsubseteq P_2^-$) and $g = P_2(x, y)$, we have $gr(g, I) = P_1(x, y)$ (resp. $gr(g, I) = P_1(y, x)$)

We are now ready to define the algorithm PerfectRef.

Algorithm PerfectRef$(q, \mathcal{T})$
Input: conjunctive query q of arity n, *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}$
Output: FOL query q_r
$P := \{q\}$;
repeat
  $P' := P$;
  for each $q \in P'$ **do**
  (a) **for each** g in q **do**
      for each PI I in $\mathcal{T}$ **do**
        if I is applicable to g
        then $P := P \cup \{\, q[g/gr(g, I)] \,\}$;
  (b) **for each** g_1, g_2 in q **do**
      if g_1 and g_2 unify
      then $P := P \cup \{\tau(reduce(q, g_1, g_2))\}$;
until $P' = P$;
let q_r be a query in P
for each $q \in P$ **do**
  $body_{q_r} = body_{q_r} \vee body_q$;
return q_r
end

In the algorithm, $q[g/g']$ denotes the query obtained from q by replacing the atom g with a new atom g'.

Informally, the algorithm first reformulates the atoms of each (conjunctive) query $q \in P'$, and produces a new (conjunctive) query for each atom reformulation (step (a)). Roughly speaking, PIs are used as rewriting rules, applied from right to left, that allow to compile away in the reformulation the knowledge of $\mathcal{T}$ that is relevant for answering q.

At step (b), for each pair of atoms g_1, g_2 that unify, the algorithm computes the query $q' = reduce(q, g_1, g_2)$, by applying to q the *most general unifier* between g_1 and g_2. Due to the unification, variables that were 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)). Function τ applied to q' replaces with _ each unbound variable in q'.

Finally, the for cycle before the end of the algorithm transforms the set of conjunctive queries P into a FOL query (union of conjunctive queries) q_r.

Lemma 7 *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a satisfiable DL-Lite$_\mathcal{R}$ knowledge base, let q be a conjunctive query over $\mathcal{K}$, and let $\vec{a}$ be a tuple of constants occurring $\mathcal{A}$. Then, $\mathcal{K} \models q(\vec{a})$ iff $\vec{a}^{I_\mathcal{A}} \in q_r^{I_\mathcal{A}}$, where $q_r = \mathsf{PerfectRef}(\mathcal{T}, q)$ and $I_\mathcal{A}$ is the minimal interpretation of $\mathcal{A}$.*

Finally, we are able to give the algorithm FOL-Reduction.

Algorithm FOL-Reduction$(q, \mathcal{T})$
Input: conjunctive query q of arity n, *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}$
Output: FOL query q_1
$\mathcal{T} = \mathsf{Normalize}(\mathcal{T})$;
let $\mathcal{T}_{PI}$ be the TBox obtained from $\mathcal{T}$ by dropping all NIs;
$\mathcal{T}_{NI} = T \backslash T_{PI}$;
$q_c = \mathsf{Consistent}(\mathcal{T}_{NI}, q)$;
$q_r = \mathsf{PerfectRef}(\mathcal{T}_{PI}, q)$;
let q_1 be a FOL query such that
    $body_{q_c} = body_{q_c} \vee body_{q_r}$;
return q_c
end

It should be easy to see that, posed $q_1 = \mathsf{FOL\text{-}Reduction}(\mathcal{T}, q)$, from Lemma 5, Lemma 6 and Lemma 7, it follows that given an ABox $\mathcal{A}$, and a tuple $\vec{a}$ of constants of $\mathcal{A}$, $\vec{a} \in q_1^{\mathcal{I}_\mathcal{A}}$ iff $(\mathcal{T}, \mathcal{A}) \models q(\vec{a})$, thus proving the thesis. □

Other logics allowing for different usages of qualified existential quantification will be analyzed in the next section.

4 FOL-reducibility of *DLR-Lite*

In this section we show that we can extend the FOL-reducibility results of Theorems 1 and 2 to the case where we allow for the presence of n-ary relations, similar to the DL *DLR* [12]. Given an interpretation $\mathcal{I}$, an n-ary relation R is interpreted as an n-ary relation $R^\mathcal{I}$ over $\Delta^\mathcal{I}$. For each of the DLs presented in Section 2, we introduce now a corresponding variant in which we allow for n-ary relations instead of (binary) roles. In the following, R denotes an n-ary relation, and we use $\vec{o}$ to denote an n-tuple of objects, and $\vec{o}[i]$ to denote the i-th component of $\vec{o}$.

The DL *DLR-Lite*$_{core}$ is obtained from *DL-Lite*$_{core}$ by replacing both in Cl and in Cr the construct $\exists R$ with $\exists i{:}R$, where R is an n-ary relation and $i \in \{1, \ldots, n\}$. Such a construct denotes the projection of the relation denoted by R on its i-th component. Formally, for an interpretation $\mathcal{I}$, we define $(\exists i{:}R)^{\mathcal{I}} = \{\vec{o}[i] \mid \vec{o} \in R^{\mathcal{I}}\}$.

DLR-Lite$_{\mathcal{F}}$, analogously to *DL-Lite*$_{\mathcal{F}}$, is obtained from *DLR-Lite*$_{core}$ by additionally allowing in the TBox for assertions of the form (*funct* $i{:}R$), stating the functionality of the i-th component of R. Formally, an interpretation $\mathcal{I}$ is a model of such an assertion if $\vec{o_1}, \vec{o_2} \in R^{\mathcal{I}}$ with $\vec{o_1}[i] = \vec{o_2}[i]$ implies $\vec{o_1}[j] = \vec{o_2}[j]$ for all $j \in \{1, \ldots, n\}$.

DLR-Lite$_{\mathcal{R}}$, analogously to *DL-Lite*$_{\mathcal{R}}$, is obtained from *DLR-Lite*$_{core}$ by replacing in Cr the constructs $\exists i{:}R$ with $\exists i{:}R.Cr_1, \ldots, Cr_n$. Such a construct denotes those objects that participate as i-th component to tuples of R in which the j-th component is an instance of Cr_j, for all $j \in \{1, \ldots, n\}$. Formally, for an interpretation $\mathcal{I}$, we define $(\exists i{:}R.Cr_1, \ldots, Cr_n)^{\mathcal{I}} = \{\vec{o}[i] \mid \vec{o} \in R^{\mathcal{I}}$ with $\vec{o}[j] \in Cr_j^{\mathcal{I}}$, for $j \in \{1, \ldots, n\}\}$. Additionally, *DLR-Lite*$_{\mathcal{R}}$ allows in the TBox for inclusion assertions between projections of relations of the form:

$$R_1[i_1, \ldots, i_k] \sqsubseteq R_2[j_1, \ldots, j_k]$$

where R_1 is an n-ary relation, $i_1, \ldots, i_k \in \{1, \ldots, n\}$, and $i_p \neq i_q$ if $p \neq q$; R_2 is an m-ary relation, $j_1, \ldots, j_k \in \{1, \ldots, m\}$, and $j_p \neq j_q$ if $p \neq q$. An interpretation $\mathcal{I}$ is a model of such an assertion if for every n-tuple of objects $\vec{o_1} \in R_1^{\mathcal{I}}$ there is an m-tuple of objects $\vec{o_2} \in R_2^{\mathcal{I}}$ such that $(\vec{o_1}[i_1], \ldots, \vec{o_1}[i_k]) = (\vec{o_2}[j_1], \ldots, \vec{o_2}[j_k])$. Analogously to Theorem 1 and 2, we can show FOL-reducibility of *DLR-Lite*$_{\mathcal{F}}$ and *DLR-Lite*$_{\mathcal{R}}$.

Theorem 8 *Query answering in DLR-Lite*$_{\mathcal{F}}$ *is FOL-reducible (and hence in* LogSpace *with respect to data complexity).*

Theorem 9 *Query answering in DLR-Lite*$_{\mathcal{R}}$ *is FOL-reducible (and hence in* LogSpace *with respect to data complexity).*

5 NLogSpace-hard DLs

In the previous section, we have pointed out the importance of languages for which query answering is FOL-reducible. In this section, we show that, as soon as we consider further, minimal extensions of *DL-Lite*$_{core}$, besides those illustrated in Section 3, we cross the boundary of LogSpace data complexity. Going beyond LogSpace data complexity means actually that we lose the property of FOL-reducibility and therefore query answering requires more powerful

engines than those available in standard relational database technology. An immediate consequence of this fact is that we cannot take advantage anymore of data management tools and query optimization techniques of current DBMSs.

The first case of this type is when we add qualified existential quantification to Cl. The second case is when we add qualified universal quantification to Cr, and the third case is when we add qualified existential quantification to Cr, while keeping the possibility of expressing functionality constraints.

Theorem 10 *Instance checking (and hence query answering) is* NLogSpace*-hard with respect to data complexity for the cases where*

1. $Cl \rightarrow A \mid \exists R.A$
   $Cr \rightarrow A$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr$

2. $Cl \rightarrow A$
   $Cr \rightarrow A \mid \forall R.A$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr$

3. $Cl \rightarrow A$
   $Cr \rightarrow A \mid \exists R.A$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr, (funct\ R)$

Proof. For Case 1, the proof is by a LogSpace reduction from reachability in directed graphs, which is NLogSpace-complete. Let $G = (N, E)$ be a directed graph, where N is a set of nodes and $E \subseteq N \times N$ is the set of edges of G, and let s, d be two nodes in N. Reachability is the problem of checking whether there is a path in G from s to d.

We define a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where the TBox $\mathcal{T}$ is constituted by a single inclusion assertion

$$\exists P.A \ \sqsubseteq \ A$$

and the ABox $\mathcal{A}$ has as constants the nodes of G, and is constituted by the membership assertion $A(d)$, and by one membership assertion $P(n, n')$ for each edge $(n, n') \in E$. It is easy to see that $\mathcal{K}$ can be constructed in LogSpace from G, s, and d. We show that there is a path in G from s to d if and only if $\mathcal{K} \models A(s)$.

"$\Leftarrow$" Suppose there is no path in G from s to d. We construct a model $\mathcal{I}$ of $\mathcal{K}$ such that $s^{\mathcal{I}} \notin A^{\mathcal{I}}$. Consider the interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} = N$, $n^{\mathcal{I}} = n$ for each $n \in N$, $P^{\mathcal{I}} = E$, and $A^{\mathcal{I}} = \{\, n \mid$ there is a path in G from n to $d \,\}$. We show that $\mathcal{I}$ is a model of $\mathcal{K}$. By construction, $\mathcal{I}$ satisfies all membership assertions

$P(n, n')$ and the membership assertion $A(d)$. Consider an object $n \in (\exists P.A)^{\mathcal{I}}$. Then there is an object $n' \in A^{\mathcal{I}}$ such that $(n, n') \in P^{\mathcal{I}}$. Then, by definition of $\mathcal{I}$, there is a path in G from n' to d, and $(n, n') \in E$. Hence, there is also a path in G from n to d and, by definition of $\mathcal{I}$, we have that $n \in A^{\mathcal{I}}$. It follows that also the inclusion assertion $\exists P.A \sqsubseteq A$ is satisfied in $\mathcal{I}$.

"$\Rightarrow$" Suppose there is a path in G from a node n to d. We prove by induction on the length k of such a path that $\mathcal{K} \models A(n)$. Base case: $k = 0$, then $n = d$, and the claim follows from $A(d) \in \mathcal{A}$. Inductive case: suppose there is a path in G of length $k-1$ from n' to d and $(n, n') \in E$. By the inductive hypothesis, $\mathcal{K} \models A(n')$, and since by definition $P(n, n') \in \mathcal{A}$, we have that $\mathcal{K} \models \exists P.A(n)$. By the inclusion assertion in $\mathcal{T}$ it follows that $\mathcal{K} \models A(n)$.

For Case 2, the proof follows from Case 1 and the observation that an assertion $\exists P.A_1 \sqsubseteq A_2$ is logically equivalent to the assertion $A_1 \sqsubseteq \forall P^-.A_2$, and that we can get rid of inverse roles by inverting the edges of the graph represented in the ABox.

For Case 3, the proof is again by a LogSpace reduction from reachability in directed graphs, and is based on the idea that an assertion $\exists P.A_1 \sqsubseteq A_2$ can be simulated by the assertions $A_1 \sqsubseteq \exists P^-.A_2$ and $(\mathit{funct}\ P^-)$. Moreover, the graph can be encoded using only functional roles, and we can again get rid of inverse roles by inverting edges.

More precisely, Let $G = (N, E)$ be a directed graph and consider the problem of reachability in G between nodes s and d. We define the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where the TBox $\mathcal{T}$ is constituted by the inclusion assertions

$$A \sqsubseteq \exists P_1.B \qquad B \sqsubseteq \exists P_1.B \qquad B \sqsubseteq \exists P_2.A \qquad (\mathit{funct}\ P_1) \qquad (\mathit{funct}\ P_2)$$

and the ABox $\mathcal{A}$ makes use of the nodes in N and the edges in E as constants. Consider a node n of G, and let $e_1, \ldots, e_k$ be all edges of G that have n as their target (i.e., such that $e_i = (n_i, n)$ for some node n_i), taken in some arbitrarily chosen order. Then the ABox $\mathcal{A}$ contains the following membership assertions:

- $P_1(n, e_1)$, and $P_1(e_i, e_{i+1})$ for $i \in \{1, \ldots, k-1\}$,
- $P_2(e_i, n_i)$, where $e_i = (n_i, n)$, for $i \in \{1, \ldots, k-1\}$.

Additionally, $\mathcal{A}$ contains the membership assertion $A(d)$. Notice that the assertions in the ABox do not violate the functionality assertions in the TBox. Again, it is easy to see that $\mathcal{K}$ can be constructed in LogSpace from G, s, and d. We show that there is a path in G from s to d if and only if $\mathcal{K} \models A(s)$.

"$\Leftarrow$" Suppose there is no path in G from s to d. We construct a model $\mathcal{I}$ of $\mathcal{K}$ such that $s^{\mathcal{I}} \notin A^{\mathcal{I}}$. Consider the interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} = \{o\} \cup N \cup E$, and in which each constant of the ABox is interpreted as itself, $P_1^{\mathcal{I}}$ and $P_2^{\mathcal{I}}$ contain all pairs of nodes directly required by the ABox assertions, $A^{\mathcal{I}}$ contains each node

n such that there is a path in G from n to d, and $B^{\mathcal{I}}$ contains all edges (i, j) such that there is a path in G from j to d. To satisfy the assertion $A \sqsubseteq \exists P_1.B$ for those objects $n \in A^{\mathcal{I}}$ that have no outgoing P_1 edge forced by the ABox (i.e., that have no incoming edge in G), we set $o \in B^{\mathcal{I}}$, $(n, o) \in P_1^{\mathcal{I}}$, and $(o, o) \in P_1^{\mathcal{I}}$. We use o in a similar way to satisfy the assertions $B \sqsubseteq \exists P_1.B$ and $B \sqsubseteq \exists P_1.A$. Note that in this way the functionality assertions are not violated. It is easy to see that $\mathcal{I}$ is a model of $\mathcal{K}$, and since there is no path in G from s to d, we have that $s \notin A^{\mathcal{I}}$.

"$\Rightarrow$" Suppose there is a path in G from a node n to d. We prove by induction on the length ℓ of such a path that $\mathcal{K} \models A(n)$. Base case: $\ell = 0$, then $n = d$, and the claim follows from $A(d) \in \mathcal{A}$. Inductive case: suppose there is a path in G of length $\ell - 1$ from j to d and $(n, j) \in E$. Let $n_1, \ldots, n_h$ be the nodes of G such that $(n_i, j) \in E$, up to $n_h = n$ and in the same order used in the construction of the ABox. By the inductive hypothesis, $\mathcal{K} \models A(j)$, and by the assertion $A \sqsubseteq \exists P_1.B$, functionality of P_1, and the ABox assertion $P_1(j, (n_1, j))$, we obtain that $\mathcal{K} \models B((n_1, j))$. Exploiting $B \sqsubseteq \exists P_1.B$, functionality of P_1, and the ABox assertion $P_1((n_i, j), (n_{i+1}, j))$, we obtain by induction on h that $\mathcal{K} \models B((n_h, j))$. Finally, by $B \sqsubseteq \exists P_2.A$, functionality of P_2, and the ABox assertion $P_2((n_h, j), n_h)$, we obtain that $\mathcal{K} \models A(n_h)$, i.e., $\mathcal{K} \models A(n)$.

□

Note that all the above "negative" results hold for instance checking already, i.e., for the simplest queries possible. Also, note that in all three cases, we are considering extensions to a minimal subset of *DL-Lite*$_{core}$ in order to get NLogSpace-hardness.

6 PTime-hard DLs

Next we show that if we consider further extensions to the logics mentioned in Theorem 10, we get even stronger complexity results. In particular, we consider five different cases where query answering (actually, instance checking already) becomes PTime-hard in data complexity.

Note that the PTime-hardness result basically means that we need at least the power of full Datalog to answer queries in these cases.

Theorem 11 *Instance checking (and hence query answering) is* PTime*-hard with respect to data complexity for the cases where*

1. $Cl \rightarrow A \mid \exists R.A$
   $Cr \rightarrow A \mid \exists P$
   $R \rightarrow P \mid P^-$
   TBox assertions: $Cl \sqsubseteq Cr$

2. $Cl \rightarrow A$
   $Cr \rightarrow A \mid \exists R.A$
   $R \rightarrow P \mid P^-$
   TBox assertions: $Cl \sqsubseteq Cr, (\mathit{funct}\ R)$

3. $Cl \rightarrow A \mid \exists R.A$
   $Cr \rightarrow A \mid \exists R.A$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr, (\mathit{funct}\ R)$

Proof. For Case 1, we reduce the emptiness problem of context-free grammars to query answering over such DL-KBs. Let $G = \langle \mathcal{V}_N, \mathcal{V}_T, S, \mathcal{P} \rangle$ be a context-free grammar (the non-terminal symbol S is the axiom of G). Without loss of generality, we can assume that each production rule has at most two nonterminal symbols in its right-hand side, since each rule with more than two nonterminal symbols in its right-hand side can be linearly transformed into an equivalent set of production rules with at most two nonterminal symbols in their right-hand side. Let $\mathcal{L}(G)$ be the language generated by G.

Given a production rule R, we denote by $\mathit{Left}(R)$ the nonterminal symbol occurring in the left-hand side of R, and denote by $\mathit{Right}(R)$ the set of nonterminal symbols occurring in the right-hand side of R.

We define the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

- The TBox $\mathcal{T}$ is constituted by the following inclusion assertions:

$$\begin{aligned} \exists L.D \sqsubseteq D \quad & (I1) \\ \exists R.D \sqsubseteq A_1 \quad & (I2) \\ \exists R.A_1 \sqsubseteq A_2 \quad & (I3) \\ \exists P.A \sqsubseteq D \quad & (I4) \\ \exists P^-.A_2 \sqsubseteq A \quad & (I5) \\ A_1 \sqsubseteq \exists P \quad & (I6) \end{aligned}$$

- The ABox $\mathcal{A}$ is constructed in the following way.

```
begin
  A := ∅;
  j = 1;
  for each N ∈ V_N do
  begin
    i=1;
    for each production rule PR in P such that Left(PR) = N do
    begin
      A := A ∪ {L(n_i, n_{i+1})};
```

```
        if Right(PR) = ∅
        then A := A ∪ {D(N_i)}
        else if Right(PR) = {B}
        then begin A := A ∪ {A_1(j), R(N_i, j), L(j, B_1)}; j := j + 1 end
        else if Right(PR) = {B, C}
        then begin
          A := A ∪ {R(N_i, j), L(j, B_1), R(j, j + 1), L(j + 1, C_1)};
          j := j + 2
        end;
        i := i + 1
      end
    end
  end
```

It is immediate to see that $\mathcal{A}$ is constructed in time linear in the size of $\mathcal{P}$. (Notice that for each nonterminal symbol A, the individuals $a_1, \ldots, a_k$ represent the k occurrences of A in the left-hand sides of production rules in $\mathcal{P}$, while there is a distinct (new) individual j for each right-hand side occurrence of A in $\mathcal{P}$).

Finally, let q be the query $q{:-}D(S_1)$. We prove that $\mathcal{L}(G)$ is empty iff $\mathcal{K} \models q$. More precisely, we prove that, for every nonterminal symbol $A \in \mathcal{V}_N$, $\mathcal{K} \models D(A)$ iff A generates a non-empty language in G.

($\Leftarrow$) Suppose A generates a non-empty language in G. We prove that $\mathcal{K} \models D(a_1)$. The proof is by induction on the structure of a derivation of s from A in G. Base case (direct derivation): there exists a production rule such that $\mathit{Left}(PR) = A$ and $\mathit{Right}(PR) = \emptyset$. By the above definition of $\mathcal{A}$, $D(a_1) \in \mathcal{A}$, consequently $\mathcal{K} \models D(a_1)$. Inductive case (indirect derivation): there exists a production rule such that $\mathit{Left}(PR) = A$ and each nonterminal symbol occurring in $\mathit{Right}(PR)$ generates a non-empty language in G. Suppose $\mathit{Right}(PR) = \{B, C\}$ (the case when $\mathit{Right}(PR) = \{B\}$ is analogous). By the inductive hypothesis, it follows that $\mathcal{K} \models D(b_1)$ and $\mathcal{K} \models D(c_1)$. Moreover, by definition of $\mathcal{A}$, there exist individuals i and j such that $R(a_k, i) \in \mathcal{A}$ for some k, $R(i, j) \in \mathcal{A}$, $L(i, b_1) \in \mathcal{A}$, $L(j, c_1) \in \mathcal{A}$. Since $\mathcal{K} \models D(c_1)$, from inclusion (I1) of $\mathcal{T}$ it follows that $\mathcal{K} \models D(j)$, consequently from inclusion (I2) it follows that $\mathcal{K} \models A_1(i)$, and from inclusion (I3) we have that $\mathcal{K} \models A_2(a_k)$; moreover, from $\mathcal{K} \models D(b_1)$ and inclusion (I1) it follows that $\mathcal{K} \models D(i)$, thus, from (I2) it follows that $\mathcal{K} \models A_1(a_k)$. Now, from inclusion (I6) we have that there exists individual ℓ such that $\mathcal{K} \models P(a_k, \ell)$, thus, from $\mathcal{K} \models A_2(a_k)$ and from inclusion (I5) it follows that $\mathcal{K} \models A(\ell)$, therefore $\mathcal{K} \models \exists P.A(a_k)$, and by inclusion (I4) it follows that $\mathcal{K} \models D(a_k)$.

($\Rightarrow$) Suppose that A generates a empty language in G. We prove that $\mathcal{K} \not\models D(a_1)$. The proof is by induction on the structure of G. The key property

is that, from the definition of $\mathcal{A}$ it follows that, for each (new) individual i in $\mathcal{A}$ corresponding to a right-hand side occurrence of the nonterminal symbol A, $\mathcal{K} \models D(i)$ if and only if $\mathcal{K} \models D(a_1)$, since the concept D "propagates backward" only through the role L, and by definition of $\mathcal{A}$, each new individual i (representing a right-hand side occurrence of A) is connected through role L only to the individual A.

For Case 2, the reduction (and the proof of its correctenss) is the same as in Case 1, with the exception of the TBox assertions which are the following:

$$\begin{aligned} D &\sqsubseteq \exists L^-.D && (I1)\\ D &\sqsubseteq \exists R^-.A_1 && (I2)\\ A_1 &\sqsubseteq \exists R^-.A_2 && (I3)\\ A &\sqsubseteq \exists P^-.D && (I4)\\ A_2 &\sqsubseteq \exists P.A && (I5)\\ A_1 &\sqsubseteq \exists P && (I6)\\ (\mathit{funct}\ L^-) && (I7)\\ (\mathit{funct}\ R^-) && (I8)\\ (\mathit{funct}\ P) && (I9) \end{aligned}$$

Also for Case 3, the reduction (and the proof of its correctenss) is the same as in Case 1, with the exception of the TBox assertions which are the following:

$$\begin{aligned} \exists L.D &\sqsubseteq D && (I1)\\ \exists R.D &\sqsubseteq A_1 && (I2)\\ \exists R.A_1 &\sqsubseteq A_2 && (I3)\\ \exists P.A &\sqsubseteq D && (I4)\\ A_2 &\sqsubseteq \exists P.A && (I5)\\ A_1 &\sqsubseteq \exists P && (I6)\\ (\mathit{funct}\ P) && (I7) \end{aligned}$$

□

Theorem 12 *Instance checking (and hence query answering) is* PTIME*-hard with respect to data complexity for the cases where*

1. $Cl \rightarrow A \mid \exists R.A \mid A_1 \sqcap A_2$
   $Cr \rightarrow A$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr$

2. $Cl \rightarrow A \mid A_1 \sqcap A_2$
   $Cr \rightarrow A \mid \forall R.A$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr$

3. $Cl \rightarrow A \mid A_1 \sqcap A_2$
$Cr \rightarrow A \mid \exists R.A$
$R \rightarrow P$
TBox assertions: $Cl \sqsubseteq Cr, (\mathit{funct}\ R)$

Proof. For Case 1, the proof is by a LogSpace reduction from Path System Accessibility, which is PTime-complete [15]. An instance of Path System Accessibility is defined as $PS = (N, E, S, t)$, where N is a set of nodes, $E \subseteq N \times N \times N$ is an accessibility relation (we call its elements edges), $S \subseteq N$ is a set of source nodes, and $t \in N$ is a terminal node. PS consists in verifying whether t is *accessible*, where a node $n \in N$ is accessible if $n \in S$ or if there exist accessible nodes n_1 and n_2 such that $(n, n_1, n_2) \in E$.

We define the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where the TBox $\mathcal{T}$ is constituted by the inclusion assertions

$$\exists P_1.A \sqsubseteq B_1 \qquad \exists P_2.A \sqsubseteq B_2 \qquad B_1 \sqcap B_2 \sqsubseteq A \qquad \exists P_3.A \sqsubseteq A$$

and the ABox $\mathcal{A}$ makes use of the nodes in N and the edges in E as constants. Consider a node $n \in N$, and let $e_1, \ldots, e_k$ be all edges in E that have n as their first component, taken in some arbitrarily chosen order. Then the ABox $\mathcal{A}$ contains the following membership assertions:

- $P_3(n, e_1)$, and $P_3(e_i, e_{i+1})$ for $i \in \{1, \ldots, k-1\}$,
- $P_1(e_i, j)$ and $P_2(e_i, k)$, where $e_i = (n, j, k)$, for $i \in \{1, \ldots, k-1\}$.

Additionally, $\mathcal{A}$ contains one membership assertion $A(n)$ for each node $n \in S$. Again, it is easy to see that $\mathcal{K}$ can be constructed in LogSpace from PS. We show that t is accessible in PS if and only if $\mathcal{K} \models A(t)$.

"$\Leftarrow$" Suppose that t is not accessible in PS. We construct a model $\mathcal{I}$ of $\mathcal{K}$ such that $t^{\mathcal{I}} \notin A^{\mathcal{I}}$. Consider the interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} = N \cup E$, and in which each constant of the ABox is interpreted as itself, $P_1^{\mathcal{I}}$, $P_2^{\mathcal{I}}$, and $P_3^{\mathcal{I}}$ consist of all pairs of nodes directly required by the ABox assertions, $B_1^{\mathcal{I}}$ consists of all edges (i, j, k) such that j is accessible in PS, $B_2^{\mathcal{I}}$ consists of all edges (i, j, k) such that k is accessible in PS, and $A^{\mathcal{I}}$ consists of all nodes n that are accessible in PS union all edges (i, j, k) such that both j and k are accessible in PS. It is easy to see that $\mathcal{I}$ is a model of $\mathcal{K}$, and since t is not accessible in PS, we have that $t \notin A^{\mathcal{I}}$.

"$\Rightarrow$" Suppose that t is accessible in PS. We prove by induction on the structure of the derivation of accessibility that if a node n is accessible, then $\mathcal{K} \models A(n)$. Base case (direct derivation): $n \in S$, hence, by definition, $\mathcal{A}$ contains the assertion $A(n)$ and $\mathcal{K} \models A(n)$. Inductive case (indirect derivation): there exists an edge $(n, j, k) \in E$ and both j and k are accessible. By the inductive hypothesis, we have that $\mathcal{K} \models A(j)$ and $\mathcal{K} \models A(k)$. Let $e_1, \ldots, e_h$ be the edges

in E that have n as their first component, up to $e_h = (n, j, k)$ and in the same order used in the construction of the ABox. Then, by $P_1(e_h, j)$ in the ABox and the assertions $\exists P_1.A \sqsubseteq B_1$ we have that $\mathcal{K} \models B_1(e_h)$. Similarly, we get $\mathcal{K} \models B_2(e_h)$, and hence $\mathcal{K} \models A(e_h)$. By exploiting assertions $P_3(e_i, e_{i+i})$ in the ABox, and the TBox assertion $\exists P_3.A \sqsubseteq A$, we obtain by induction on h that $\mathcal{K} \models A(e_1)$. Finally, by $P_3(n, e_1)$, we obtain that $\mathcal{K} \models A(n)$.

For Cases 2 and 3, the proof follows from Case 1 and observations analogous to the ones for Theorem 10. □

7 coNP-hard DLs

Finally, we show three cases where the TBox language becomes so expressive that the data complexity of query answering goes beyond PTime (assuming PTime $\neq$ NP).

Theorem 13 *Query answering is coNP-hard with respect to data complexity for the cases where*

1. $Cl \rightarrow A \mid \neg A$
   $Cr \rightarrow A$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr$

2. $Cl \rightarrow A$
   $Cr \rightarrow A \mid A_1 \sqcup A_2$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr$

3. $Cl \rightarrow A \mid \forall R.A$
   $Cr \rightarrow A$
   $R \rightarrow P$
   TBox assertions: $Cl \sqsubseteq Cr$

Proof. In all three cases, the proof is an adaptation of the proof of coNP-hardness of instance checking for $\mathcal{ALE}$ presented in [14]. In the following, we first consider case 2.

coNP-hardness of query answering is proved by a reduction from $2+2$-CNF unsatisfiability (which is showed to be coNP-complete in [14]). A $2+2$-CNF formula on an alphabet P is a CNF formula in which each clause has exactly four literals: two positive and two negative ones, where the propositional letters are elements of $P \cup \{true, false\}$. Given a $2+2$-CNF formula $F = C_1 \wedge \ldots \wedge C_n$, where $C_i = L^i_{1+} \vee L^i_{2+} \vee \neg L^i_{1-} \vee \neg L^i_{2-}$, we associate with it a knowledge base

$\mathcal{K}_F = (\mathcal{T}_F, \mathcal{A}_F)$ and a query Q as follows. $\mathcal{K}_F$ has one constant ℓ for each letter L in F, one constant c_i for each clause C_i, plus two constants *true* and *false* for the corresponding propositional constants. The atomic roles of $\mathcal{K}_F$ are P_1, P_2, N_1, N_2 and the atomic concepts are O, A_t, and A_f. Then, we pose

$$
\begin{array}{rcl}
\mathcal{T}_F & = & \{O \sqsubseteq A_t \cup A_f\}, \\
\mathcal{A}_F & = & \{\ A_t(\mathit{true}), A_f(\mathit{false}) \\
 & & O(\ell^1_{1+}), O(\ell^1_{2+}), O(\ell^1_{1-}), O(\ell^1_{2-}), \\
 & & \ldots \\
 & & O(\ell^n_{1+}), O(\ell^n_{2+}), O(\ell^n_{1-}), O(\ell^n_{2-}), \\
 & & P_1(c_1, \ell^1_{1+}), P_2(c_1, \ell^1_{2+}), N_1(c_1, \ell^1_{1-}), N_2(c_1, \ell^1_{2-}), \\
 & & \ldots \\
 & & P_1(c_n, \ell^n_{1+}), P_2(c_n, \ell^n_{2+}), N_1(c_n, \ell^n_{1-}), N_2(c_n, \ell^n_{2-})\ \}, \text{ and} \\
Q & = & \{\ \mid \exists\, x, y, z, w_1, w_2 . P_1(x, y) A_f(y) P_2(x, z) A_f(z) N_1(x, w_1) A_t(w_1) N_2(x, w_2) A_t(w_2)\}.
\end{array}
$$

Intuitively, the membership to the extension of A_f or A_t corresponds to the truth values *true* and *false* respectively and checking $\mathcal{K}_F \models Q$ (i.e., the query evaluates to true in $\mathcal{K}_F$) corresponds to checking whether in every truth assignment for the formula F there exists a clause whose positive literals are interpreted as false, and whose negative literals are interpreted as true, i.e., a clause that is not satisfied. Note that the ABox $\mathcal{A}_F$ contains the assertions $A_t(\mathit{true})$ and $A_f(\mathit{false})$ in order to guarantee that in each model $\mathcal{I}$ of $\mathcal{K}_F$ the constants *true* and *false* are in the extension of (possibly both) $A_t^{\mathcal{I}}$ and $A_f^{\mathcal{I}}$, respectively.
Now, it remains to prove that the formula F is unsatisfiable if and only if $\mathcal{K}_F \models Q$.

"$\Leftarrow$" Suppose that F is unsatisfiable. Consider a model $\mathcal{I}$ of $\mathcal{K}_F$ (which always exists since $\mathcal{K}_F$ is always satisfiable), and let $\delta_{\mathcal{I}}$ be the truth assignment for F such that $\delta_{\mathcal{I}}(\ell) = \mathit{true}$ iff $\ell^{\mathcal{I}} \in A^{\mathcal{I}}$, for every letter ℓ in F (and corresponding constant in $\mathcal{K}_F$). Since F is unsatisfiable, there exists a clause C_i that is not satisfied by $\delta_{\mathcal{I}}$, and therefore $\delta_{\mathcal{I}}(L^i_{1+}) = \mathit{false}$, $\delta_{\mathcal{I}}(L^i_{2+}) = \mathit{false}$, $\delta_{\mathcal{I}}(L^i_{1-}) = \mathit{true}$ and $\delta_{\mathcal{I}}(L^i_{2-}) = \mathit{true}$. It follows that in $\mathcal{K}_F$ the interpretation of the constants related to c_i through the roles P_1 and P_2 is not in $A_t^{\mathcal{I}}$, and consequently is in the $A_f^{\mathcal{I}}$, and the interpretation of constants related to c_i through the roles N_1 and N_2 is in $A_t^{\mathcal{I}}$. Thus, there exists a substitution σ which assigns variables in Q to constants in $\mathcal{K}_F$ in such a way that $\sigma(Q)$ evaluates to true in $\mathcal{I}$ (notice that this holds even if the propositional constants *true* or *false* occur in F). Therefore, since this argument holds for each model $\mathcal{I}$ of $\mathcal{K}_F$, we can conclude that $\mathcal{K}_F \models Q$.

"$\Rightarrow$" Suppose that F is satisfiable, and let δ be a truth assignment satisfying F. Let $\mathcal{I}_\delta$ be the interpretation for $\mathcal{K}_F$ defined as follows:

- $O^{\mathcal{I}_\delta} = \{\ell^{\mathcal{I}_\delta} \mid \ell \text{ occurs in } F\}$,

- $A_t^{\mathcal{I}_\delta} = \{\ell^{\mathcal{I}_\delta} \mid \delta(\ell) = \mathit{true}\} \cup \{\mathit{true}\}$,
- $A_f^{\mathcal{I}_\delta} = \{\ell^{\mathcal{I}_\delta} \mid \delta(\ell) = \mathit{false}\} \cup \{\mathit{false}\}$,
- $\rho^{\mathcal{I}_\delta} = \{(a^{\mathcal{I}_\delta}, b^{\mathcal{I}_\delta}) \mid \rho(a,b) \in \mathcal{A}_F\}$ for $\rho = P_1, P_2, N_1, N_2$.

It is easy to see that $\mathcal{I}_\delta$ is a model of $\mathcal{K}_F$. On the other hand, since F is satisfiable, for every clause in F there exists a positive literal interpreted as true or a negative literal interpreted as false. It follows that for every constant c_i, there exists either a role (P_1 or P_2) that relates c_i to a constant whose interpretation is in $A_t^{\mathcal{I}_\delta}$ or there exists a role (N_1 or N_2) that relates c_i to a constant whose interpretation is in $A_f^{\mathcal{I}_\delta}$. Since the query Q is evaluated to true in $\mathcal{I}_\delta$ only if there exists at least a constant c_i in $\mathcal{K}_F$ such that the interpretations of the constants related to c_i by roles P_1 and P_2 are both in $A_f^{\mathcal{I}_\delta}$ and the interpretations of the constants related to c_i by roles N_1 and N_2 are both in $A_t^{\mathcal{I}_\delta}$, it follows that the query Q evaluates to false in $\mathcal{I}_\delta$ and therefore $\mathcal{K}_F \not\models Q$.

Proofs for cases 1 and 3 are obtained by analogous reductions from 2+2-CNF unsatisfiability. More precisely, for case 1 the knowledge base $\mathcal{K}_F = (\mathcal{T}_F, \mathcal{A}_F)$ has the same constants and the same atomic roles as for case 2, and has only the atomic concepts A_t and A_f. Then, $\mathcal{T}_F = \{\neg A_t \sqsubseteq A_f\}$ and $\mathcal{A}_F$ is as for case 2 but without the assertions involving the concept O. Finally, the query Q is as for case 2. For case 3, $\mathcal{K}_F$ has the same constants as for cases 1 and 2, the same atomic roles as for cases 1 and 2 plus the atomic role P, and the atomic concepts A and A_f. Then, $\mathcal{T}_F = \{\forall P.A \sqsubseteq A_f\}$ and $\mathcal{A}_F$ is as for case 1 but without the assertion $A_t(\mathit{true})$, which is substituted by the assertion $P(\mathit{true}, d)$, where d is a new constant not occurring elsewhere in $\mathcal{K}_F$. Finally, the query Q is as follows

$$\begin{aligned} Q \;=\; \{\ \mid \exists\, x, y, z, w_1, w_2 . & P_1(x,y) A_f(y) P_2(x,z) A_f(z) N_1(x,w_1) \\ & P(w_1,w_2) N_2(x,w_2) P(w_3,w_4)\}. \end{aligned}$$

Soundness and completeness of the above reductions can be proved as done for the reduction of case 2. We finally point out that the intuition behind the above results is that in all three cases it is possible to require a reasoning by case analysis, caused by set covering assertions. Indeed, whereas in case 2 we have explicitly asserted $O \sqsubseteq A_t \sqcup A_f$, for the other cases this can be seen by considering that A_t and A_f, and $\forall P.A$ and $\exists P$ cover the entire domain in case 1 and case 3, respectively. □

8 Related work

All the DLs studied in this paper are fragments of expressive DLs with assertions and inverses studied in the 90's (see [8] 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 [1], Racer [4] and Pellet [2] have been developed. Indeed, one could use, off-the-shelf, a system like Racer or Pellet to perform instance checking in such DLs. Also, reasoning with conjunctive queries in these DLs has been studied (see e.g., [12, 13]), although not yet implemented in systems. Unfortunately, the known reasoning algorithms for these DLs are in 2ExpTime with respect to combined complexity, and more importantly they are not tailored towards obtaining tight complexity bounds with respect to data complexity (they are in ExpTime). Alternative reasoning procedures that allow for clearly isolating data complexity have recently been proposed, how they will work in practice still needs to be understood. A coNP upper bound for data complexity of instance checking in the expressive DL $\mathcal{SHIQ}$ has been shown by making use of a reduction to Disjunctive Datalog and then exploiting resolution [18, 19]. It remains open whether such a technique can be extended to deal efficiently with conjunctive queries. In [21], making use of an algorithm based on tableaux, a coNP, upper-bound with respect to data complexity is given for a DL with arbitrary inclusion assertions, but lacking inverse roles. Recently, building on such techniques, coNP-completeness of answering conjunctive queries for $\mathcal{SHIQ}$, which includes inverse roles, and number restrictions (that generalize functionality) has been shown [22]. It is interesting to observe that the results in this paper (Theorem 13) tell us that we get coNP-completeness already for very small fragments of $\mathcal{SHIQ}$.

In [19], a fragment of $\mathcal{SHIQ}$, called Horn-$\mathcal{SHIQ}$, which subsumes both *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$, is studied and a PTime upper bound in data complexity for instance checking is shown. The results in the current paper (Theorem 11) tell us that instance checking in Horn-$\mathcal{SHIQ}$ is also PTime-hard. Indeed, Horn-$\mathcal{SHIQ}$ allows for qualified existential quantification $\exists P.A$ in both sides of inclusion assertions and (an extended form) of functionality restrictions.

DL-Lite$_{\mathcal{R}}$ captures (the DL-subset of) RDFS extended with participation constraints (i.e., inclusion assertions with $\exists R$ on the right-hand side). Hence, query answering over an RDFS ontology, even extended with participation constraints, is FOL-reducible. Finally, if we move from RDFS to DLP [16], query answering becomes PTime-hard, since DLP is a superset of the DL in case 1 of Theorem 12.

9 Conclusions

We have presented first fundamental results on the data complexity (complexity with respect to the size of the ABox only) of query answering in DLs. In particular, we have concentrated on the FOL-reducibility boundary of the problem, based on the observation that, when we go above this boundary, query answering

Cl	Cr	$\mathcal{F}$	$\mathcal{R}$	Data complexity of query answering
DL-Lite$_{\mathcal{F}}$		$\surd$	–	in LogSpace
DL-Lite$_{\mathcal{R}}$		–	$\surd$	in LogSpace
DLR-Lite$_{\mathcal{F}}$		$\surd$	–	in LogSpace
DLR-Lite$_{\mathcal{R}}$		–	$\surd$	in LogSpace
$A \mid \exists P.A$	A	–	–	NLogSpace-hard
A	$A \mid \forall P.A$	–	–	NLogSpace-hard
A	$A \mid \exists P.A$	$\surd$	–	NLogSpace-hard
$A \mid \exists P.A \mid \exists P^{-}.A$	$A \mid \exists P$	–	–	PTime-hard
A	$A \mid \exists P.A \mid \exists P^{-}.A$	$\surd$	–	PTime-hard
$A \mid \exists P.A$	$A \mid \exists P.A$	$\surd$	–	PTime-hard
$A \mid \exists P.A \mid A_1 \sqcap A_2$	A	–	–	PTime-hard
$A \mid A_1 \sqcap A_2$	$A \mid \forall P.A$	–	–	PTime-hard
$A \mid A_1 \sqcap A_2$	$A \mid \exists P.A$	$\surd$	–	PTime-hard
$A \mid \neg A$	A	–	–	coNP-hard
A	$A \mid A_1 \sqcup A_2$	–	–	coNP-hard
$A \mid \forall P.A$	A	–	–	coNP-hard

Legenda: A (possibly with subscript) = atomic concept, P = atomic role, Cl/Cr = left/right-hand side of inclusion assertions, $\mathcal{F}$ = functionality assertions allowed, $\mathcal{R}$ = role/relationship inclusions allowed. NLogSpace and PTime hardness results hold already for instance checking.

Figure 1: Data Complexity of Query Answering in Description Logics

is no longer expressible as a first-order logic formula (and hence an SQL query) over the data. The results provided in this paper are summarized in Figure 1.

We are currently following several directions to continue the work reported in this paper. First, we conjecture that for all NLogSpace and PTime-hardness results presented here a matching upper bound holds. Second, although here we focused on data complexity only, we are also working on characterizing the complexity of query answering with respect to the size of the TBox, with respect to the size of the query, and with respect to combined complexity. Finally, while in this paper we considered conjunctive queries, our general goal is to come up with a clear picture of how the complexity of query answering is influenced not only by different TBox languages, but also by different query languages.

References

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

[2] http://www.mindswap.org/2003/pellet/.

[3] http://www.omg.org/uml/.

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

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

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

[7] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[8] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[9] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.

[10] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-147/, 2005.

[11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[12] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[13] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[14] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability — A guide to NP-completeness*. W. H. Freeman and Company, San Francisco (CA, USA), 1979.

[16] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of the 12th Int. World Wide Web Conf. (WWW 2003)*, pages 48–57, 2003.

[17] J. Heflin and J. Hendler. A portrait of the semantic web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[18] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}$-description logic to disjunctive datalog programs. In *Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004)*, 2004.

[19] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.

[20] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[21] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[22] M. M. Ortiz de la Fuente, D. Calvanese, T. Eiter, and E. Franconi. Data complexity of answering conjunctive queries over $\mathcal{SHIQ}$ knowledge bases. Technical report, Faculty of Computer Science, Free University of Bozen-Bolzano, July 2005. Also available as CORR technical report at `http://arxiv.org/abs/cs.LO/0507059/`.

[23] M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.