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
39100 Bolzano, Italy
calvanese@inf.unibz.it

[2] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113
00198 Roma, Italy
lastname@dis.uniroma1.it

Abstract

In this paper we study data complexity of answering conjunctive queries over DL knowledge bases constituted by an ABox and a TBox. In particular, we characterize the LogSpace boundary (over the data) of the problem. This boundary is particularly meaningful because, when we go above it, query answering is not expressible as a first-order logic formula (and hence an SQL query) over the data. Within the LogSpace boundary we essentially find *DL-Lite*, a simple DL that is rich enough to express the core of UML class diagrams. The second contribution of the paper is to establish coNP-hardness of query answering with respect to data complexity for various cases of surprisingly simple DLs.

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 Systems, Data Integration Systems [11], and Semantic Web [8], data become instances of concepts in ontologies. In these contexts, data are typically very large (much larger than the intentional level of the ontologies), and query answering becomes one of the basic reasoning services. Hence, when measuring the computational complexity of query answering (and reasoning in general) the most important parameter is the size of the data.

In this paper we consider conjunctive queries (CQs) specified over Description Logics (DL) knowledge bases constituted by an ABox and a TBox, and study the data complexity of the query answering problem, i.e., the complexity of computing the answers to a CQ that are a logical consequence of the ABox and the TBox, measured with respect to the size of the ABox only. In particular, we are interested in characterizing the LogSpace boundary of the problem, i.e., finding maximally expressive DLs for which query answering can be done in LogSpace. This boundary is particularly meaningful because, when we go above it, query answering is no longer expressible as a first-order logic (FOL) formula over the data, i.e., it is not possible

to answer a CQ by first computing in LogSpace its reformulation as a first-order query, and then evaluating the reformulation over the ABox. Since first-order queries can be expressed in SQL, an important characteristic of the DLs for which query answering can be done in LogSpace is that they allow for taking advantage of Data Base Management System (DBMS) techniques for both representing data, i.e., ABox assertions, and answering queries via reformulation into SQL[1].

The contributions of the paper are the following.

- We discuss DLs for which query answering is in LogSpace. In this class, we essentially find *DL-Lite* [5], a simple DL that is rich enough to express basic ontology languages, conceptual data models (e.g., Entity-Relationship [1]), and object-oriented formalisms (e.g., basic UML class diagrams[2]). We also analyze some possible extensions of *DL-Lite* that preserve the computational complexity characterization we are interested in.

- We show that minimal additions to the languages considered above bring data complexity of query answering to NLogSpace-hardness and PTime-hardness, thus losing the possibility of reformulating queries in first-order logic. In fact, we conjecture that for such languages query answering is in PTime, and thus still feasible in practice, but the database technologies needed (Datalog engines) are not as mature as standard SQL-based DBMSs.

- Finally, we establish coNP-hardness of query answering with respect to data complexity for surprisingly simple DLs. In particular, we get intractability as soon as the logic is able to express simple forms of covering constraints.

What emerges from our complexity analysis is that *DL-Lite* is a maximal DL which allows query answering through standard database technology. In this sense, *DL-Lite* is the first DL specifically tailored for effective query answering over large amounts of data.

Actually, *DL-Lite* is a fragment of expressive DLs with assertions and inverses studied in the 90's (see [2] 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[3] and Racer[4] have been developed. Indeed, one could use, off-the-shelf, a system like Racer to perform KB satisfiability, instance checking (of concepts), and logical implication of inclusion assertions in *DL-Lite*.

The paper is organized as follows. In the next section we introduce some preliminaries which will be useful for the following discussions. In Section 3, we present DLs for which query answering is in LogSpace, and in Section 4 DLs for which query answering goes beyond LogSpace. In Section 5 we overview related work, and in Section 6 we draw some conclusions.

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

[2] `http://www.omg.org/uml/`

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

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

2 Preliminaries

We are interested in queries, and more specifically in conjunctive queries, expressed over DL knowledge bases (KBs) formed by a TBox and an ABox. The TBox is formed by a set of inclusion assertions of the form

$$B \sqsubseteq C$$

and a set of functionality assertions of the form

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

expressing the functionality of the role R.

In *DL-Lite*, we distinguish between the constructs that we allow in the concept on the left-hand side (B) and in the right-hand side (C) of inclusion assertions. Depending on the actual language that we use for B and C, we get different complexities, as shown in the next sections. As for roles R, depending on the particular language, we allow either atomic roles only (denoted by P), or atomic and inverse roles (P^-).

First, we observe that including $B_1 \sqcup B_2$ in the constructs for B on the left-hand side and $C_1 \sqcap C_2$ in the constructs for C on the right-hand side does not affect the results below, since they can be simulated by inclusion assertions as follows:

- $B_1 \sqcup B_2 \sqsubseteq C$ is equivalent to $B_1 \sqsubseteq C$ and $B_2 \sqsubseteq C$;
- $B \sqsubseteq C_1 \sqcap C_2$ is equivalent to $B \sqsubseteq C_1$ and $B \sqsubseteq C_2$.

Similarly, we can trivially add $\bot$ to the language for B on the left-hand side and $\top$ to the language for C on the right-hand side.

As for the ABox, we allow for membership assertions on atomic concepts and on atomic roles:

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

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

Given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox, we can query the knowledge base using conjunctive queries. A conjunctive query q is an expression of the form

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

where $\vec{x}$ are the so called distinguished variables that will be bound with objects in the KB, $\vec{y}$ are the non-distinguished variables, which are existentially qualified variables, and $\mathit{conj}(\vec{x}, \vec{y})$ is a conjunction of atoms of the form $A(z)$ or $P(z_1, z_2)$ where A and P are respectively atomic concept and roles and z, z_1, z_2 are either constants in the KB or variables in $\vec{x}$ or $\vec{y}$. When the query is boolean and the conjunction is constituted by a single atom $A(a)$ (resp., $P(a, b)$), with a (resp., a, b) a constant in the KB, we simply denote it by $A(a)$ (resp., $P(a, b)$).

The reasoning service we are interested in is *conjunctive query answering*: given a conjunctive query $q(\vec{x})$ with distinguished variables $\vec{x}$ and a knowledge base $\mathcal{K}$, return

all tuples $\vec{t}$ of objects in $\mathcal{K}$ that substituted to $\vec{x}$ are such that $\mathcal{K} \models q(\vec{t})$. In the following, we will simply use the term "query answering", and omit the qualification "conjunctive".

We observe that both concept satisfiability and instance checking can be seen as special cases of query answering. In particular, instance checking, i.e., logical implication of ABox assertions, can be expressed as the problem of answering conjunctive queries constituted by just one ground atom.

Finally, we will talk about data complexity of query answering. The notion of data complexity is borrowed from relational database theory [14], and in the context of DLs is defined as follows. 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 $\vec{t}$ of objects, we have that $(\mathcal{T}, \mathcal{A}) \models q(\vec{t})$. Note that neither the TBox nor the query is an input to the recognition problem. When we say that query answering for a certain DL $\mathcal{L}$ *is in* S (where S is a complexity class) *with respect to data complexity*, we mean that the corresponding recognition problem is in S. Similarly, when we say that query answering for a certain DL $\mathcal{L}$ is *S-hard with respect to data complexity*, we mean that the corresponding recognition problem is S-hard.

3 Within LogSpace

In this section we discuss several DLs for which query answering is in LogSpace.

The first DL of this kind is *DL-Lite* [5], where

$$\begin{array}{lcl} B & \rightarrow & A \mid \exists R \\ C & \rightarrow & B \mid \neg B \\ R & \rightarrow & P \mid P^- \end{array}$$

(*funct* R) is allowed

An important feature of this logic is that it is perfectly suited for representing ABox assertions managed in secondary storage by a relational DBMS. Indeed, the query answering algorithm presented in [5] is 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. Hence, the resulting union of conjunctive queries is of constant size with respect to the size of the ABox[5]. Since conjunctive queries are FOL formulas, and since the evaluation of FOL formulas can be done in LogSpace, it follows that query answering in *DL-Lite* is in LogSpace. Notably, the consistency check that is part of query answering can also be reduced to the evaluation of a set of conjunctive queries (with inequalities) over the ABox.

[5]Note that the union of conjunctive queries resulting from the expansion is polynomial in the size of the TBox. Also, each conjunctive query in such a union is linear in the size of the original query, though the total number of conjunctive queries produced by the expansion may be exponential in the size of the original query.

From a practical point of view, this is an important result. Indeed, the query generated by our expansion algorithm can be evaluated by an SQL engine over a simple relational database defined by ABox assertions, thus taking advantage of well-established query optimization strategies that we find in relational DBMSs.

Interestingly, the above method can be adapted to the case where we extend *DL-Lite* with new features. In particular, we have proved that if we extend B with conjunction, and C with $\bot$,[6] we still have the possibility of reducing query answering to FOL evaluation over the ABox. From this, we get the following result.

Theorem 1 *Query answering is in* LogSpace *with respect to data complexity for the case where*

$$\begin{array}{lcl} B & \rightarrow & A \mid \exists R \mid B_1 \sqcap B_2 \\ C & \rightarrow & A \mid \bot \mid \exists R \\ R & \rightarrow & P \mid P^- \end{array}$$

(*funct* R) *is allowed*

Proof (sketch). The query answering algorithm for this logic is obtained by extending the reformulation technique of *DL-Lite* [5], which is independent of the ABox and produces a union of conjunctive queries over the ABox. Hence, we get the LogSpace upper bound. □

Notice that *DL-Lite* and the extension mentioned in the above theorem only allow for unqualified existential quantification. One might ask what happens to the complexity of query answering if we add qualified existential quantification to the language. It turns out that, if qualified existentials are allowed in C, then query answering is still in LogSpace, provided that we get rid of functionality constraints.

Theorem 2 *Query answering is in* LogSpace *with respect to data complexity for the case where*

$$\begin{array}{lcl} B & \rightarrow & A \mid \exists R \mid B_1 \sqcap B_2 \\ C & \rightarrow & A \mid \bot \mid \exists R.C \\ R & \rightarrow & P \mid P^- \end{array}$$

(*funct* R) *is not allowed*

Proof (sketch). Again, the query answering algorithm for this logic is obtained by extending the reformulation technique of *DL-Lite*. □

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

4 Beyond LogSpace

In the previous section, we have pointed out the importance of reducing query answering over a knowledge base to FOL query evaluation over the ABox only.

[6] Note that having $\bot$ in C and $\sqcap$ on B allows for expressing $\neg B$ in C.

In this section, we show that, as soon as we consider further, minimal extensions of *DL-Lite*, besides those illustrated in Section 3, we cross the boundary of LogSpace data complexity. Going beyond LogSpace data complexity means actually that query answering requires more powerful engines than those available in standard relational database technology. An immediate consequence of this fact is that we lose the possibility of taking advantage 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 B. The second case is when we add qualified universal quantification to C, and the third case is when we add qualified existential quantification to C, while keeping the possibility of expressing functionality constraints.

Theorem 3 *Instance checking (and hence query answering) is* NLogSpace*-hard with respect to data complexity for the cases where*

1. $B \rightarrow A \mid \exists P.A$
   $C \rightarrow A$
   $R \rightarrow P$
   (*funct* R) *is not allowed*

2. $B \rightarrow A$
   $C \rightarrow A \mid \forall P.A$
   $R \rightarrow P$
   (*funct* R) *is not allowed*

3. $B \rightarrow A$
   $C \rightarrow A \mid \exists P.A$
   $R \rightarrow P$
   (*funct* R) *is allowed*

Proof (sketch). For Case 1, the proof is by a log-space 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 log-space 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 log-space 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 (*funct* P^-). Moreover, the graph can be encoded using only functional roles, and we can again get rid of inverse roles by inverting edges. □

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 subset of *DL-Lite* in order to get NLogSpace-hardness. For example, in the first case, C has the simplest form possible, and neither inverse roles nor functionality constraints are used in the reduction.

As for the upper bound of query answering in the above logics, we point out that it is open whether query answering can be done in NLogSpace.

Next we show that if we consider further extensions to the logics mentioned in Theorem 3, we get even stronger complexity results. In particular, we first consider three cases where query answering (actually, instance checking already) becomes PTime-hard in data complexity. The three cases are obtained from the ones in Theorem 3 by adding conjunction on the left-hand side of inclusion assertions. 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 4 *Instance checking (and hence query answering) is* PTime*-hard with respect to data complexity for the cases where*

1. $B \rightarrow A \mid \exists P.A \mid B_1 \sqcap B_2$
   $C \rightarrow A$
   $R \rightarrow P$
   (*funct* R) *is not allowed*

2. $B \rightarrow A \mid B_1 \sqcap B_2$
   $C \rightarrow A \mid \forall P.A$
   $R \rightarrow P$
   (*funct* R) *is not allowed*

3. $B \rightarrow A \mid B_1 \sqcap B_2$
   $C \rightarrow A \mid \exists P.A$
   $R \rightarrow P$
   (*funct* R) *is allowed*

Proof (sketch). For Case 1, the proof is by a log-space reduction from Path System Accessibility, which is PTIME-complete [7]. 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 3. □

We conjecture that in all the cases mentioned in both Theorem 3 and Theorem 4, query answering can be actually done in PTime.

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

1. $B \rightarrow A \mid \neg A$
   $C \rightarrow A$
   $R \rightarrow P$
   (*funct* R) *is not allowed*

2. $B \rightarrow A$
   $C \rightarrow A \mid A_1 \sqcup A_2$
   $R \rightarrow P$
   (*funct* R) *is not allowed*

3. $B \rightarrow A \mid \forall P.A$
   $C \rightarrow A$
   $R \rightarrow P$
   (*funct* R) *is not allowed*

Proof (sketch). In all three cases, the proof is an adaptation of the proof of coNP-hardness of instance checking for $\mathcal{ALE}$ in [6], by re-expressing the $\mathcal{ALE}$ concept in that proof as a conjunctive query. □

In all the above cases, query answering becomes polynomially intractable with respect to data complexity. These cases are therefore particularly challenging, especially in those situations where the size of the ABox is substantial. Actually, the feasibility of query answering in theses cases is seriously hampered.

5 Related work

Reasoning with conjunctive queries in expressive DLs with assertions and inverse roles has been recently studied (see e.g., [3, 4]), 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).

The problem of answering conjunctive queries over a DL knowledge base has also been considered in [12]. Making use of an algorithm based on tableaux, a coNP, and

B	C	R	(*funct* R)	Complexity
$A \mid \exists R \mid B_1 \sqcap B_2$	$A \mid \bot \mid \exists R$	$P \mid P^-$	*allowed*	in LogSpace
$A \mid \exists R \mid B_1 \sqcap B_2$	$A \mid \bot \mid \exists R.C$	$P \mid P^-$	*not allowed*	in LogSpace
$A \mid \exists P.A$	A	P	*not allowed*	NLogSpace-hard
A	$A \mid \forall P.A$	P	*not allowed*	NLogSpace-hard
A	$A \mid \exists P.A$	P	*allowed*	NLogSpace-hard
$A \mid \exists P.A \mid B_1 \sqcap B_2$	A	P	*not allowed*	PTime-hard
$A \mid B_1 \sqcap B_2$	$A \mid \forall P.A$	P	*not allowed*	PTime-hard
$A \mid B_1 \sqcap B_2$	$A \mid \exists P.A$	P	*allowed*	PTime-hard
$A \mid \neg A$	A	P	*not allowed*	coNP-hard
A	$A \mid A_1 \sqcup A_2$	P	*not allowed*	coNP-hard
$A \mid \forall P.A$	A	P	*not allowed*	coNP-hard

Legenda: A (possibly with subscript)= atomic concept, P= atomic role, B (possibly with subscript) = left-hand side of TBox inclusion assertions, C = right-hand side of TBox inclusion assertions, R= auxiliary symbol.

Figure 1: Data Complexity of Query Answering in Description Logics

thus optimal (cf., Theorem 5), upper-bound with respect to data complexity is given for a DL with arbitrary inclusion assertions, but lacking inverse roles. Instance checking in $\mathcal{SHIQ}$, a DL with inverse roles and role hierarchies, is addressed in [10], where a coNP upper-bound is given. The paper presents also a fairly general sub-language of $\mathcal{SHIQ}$, subsuming *DL-Lite*, for which instance checking stays polynomial. The results are shown by making use of a reduction to Disjunctive Datalog and then exploiting resolution [9]. However, the latter techniques seem not to be easily generalizable to the problem of answering conjunctive queries (as opposed to queries constituted by a single atom) without incurring in a blowup that is exponential in the size of the data. In [13] the above results have been generalized, and a coNP algorithm for answering conjunctive queries in $\mathcal{SHIQ}$ has been devised. It is interesting to observe that, from the point of view of data complexity, once we allow for the rather simple combinations of constructs of the DLs in Theorem 5, we already obtain the same coNP-completeness bound as for the quite expressive DL $\mathcal{SHIQ}$.

6 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 LogSpace boundary of the problem, based on the observation that, when we go above this boundary, query answering 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 working on finding tighter upper bounds in all the cases that have been left open by our analysis.

Although we focused on data complexity, 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.

While in this paper we considered conjunctive queries, our general goal is to come up with a clear picture on how the complexity of query answering is influenced not only by different TBox languages, but also by different query languages.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. DL-Lite: Practical reasoning for rich DLs. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-104/`, 2004.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.