Ontology Based Query Answering
The Story So Far

Magdalena Ortiz

Institute of Information Systems, Vienna University of Technology
ortiz@kr.tuwien.ac.at

Abstract. Databases where relations have arity at most two, often called graph databases, play a prominent role in many fields. Ontology languages based on Description Logics (DLs) have been advocated to enrich such repositories (known as ABoxes in DL jargon) with ontological information. Then, in Ontology Based Query Answering (OBQA), one is interested in answering user queries over the data while taking into account the ontology. However, the ontological layer has a significant effect on the query answering problem, and OBQA algorithms are usually more involved than their counterparts in plain (graph) databases.
The development of OBQA algorithms and their implementation has been the goal of significant research efforts in the DL community in the last decade. Here we review some of the achieved results. We discuss the main challenges to be overcome when the ontological knowledge is expressed in different DLs, and when different query languages are considered. We give an overview of some of the algorithms developed so far, and the computational complexity of the problem for the different combinations of DLs and query languages.

1 Introduction

In many application areas, data can be naturally modeled and stored using only unary and binary relations. Indeed, *graph databases*, which can be seen as databases where only relations of these arities occur, have gained much attention in the last years and are being applied in popular fields like the Semantic Web. *Description Logics* are a family of languages for knowledge representation and reasoning whose vocabulary is based on unary and binary relations, known as *concepts* and *roles* in DL jargon [3]. This makes DLs very well suited for describing and reasoning about graph databases. Indeed, a DL data repository, known as *ABox*, is essentially a graph database. Enriching an ABox with a DL ontology allows to capture the intended semantics of an application domain by defining relations between the concepts and roles in the data source, and possibly extending the vocabulary with additional terms. Then one can query the data using the extended vocabulary, and taking into account all the relations implied by the ontology. This is essentially the idea underlying *Ontology Based Data Access (OBDA)* [6], although the OBDA setting is more general: rather than simply having an ABox over the same vocabulary as the ontology, in OBDA arbitrary relational data sources may be related to the concepts and roles in the ontology via mappings. The simplified setting we consider here is suitable for focusing on query evaluation, a crucial aspect of OBDA that has received much attention in the DL community over the last decade. Hence, here we disregard the mappings and simply assume that input queries are to be evaluated over a DL ABox (i.e., a graph database), in the presence of an ontology. This problem in often called *ontology based query answering (OBQA)* or simply *ontological query answering.*

Example 1. For example, consider a simple graph database containing data from the *Mathematics Genealogy Project (MGP)*[1], like the one depicted in Figure 1.The labels inside the nodes are the names of the constants, while the labels outside are the concepts (unary relations) each node satisfies, and the labels on the arcs are the roles (binary relations) that hold between two objects.

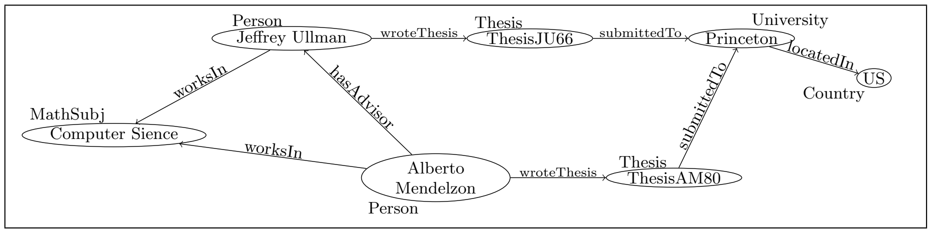

Fig. 1: Example graph database of the Mathematics Genealogy Project

A DL ontology describing this domain is given in Figure 2. The first axiom introduces the term *CompScientist* and ensures that everyone who works on the *CompScience* field is a *CompScientist*. Axiom (2) states that every PhD holder must have an advisor who is himself a PhD holder. Axioms (3,4) together ensure that PhD holders are exactly those people that have written and submitted a (PhD) thesis. Finally, axioms (5,6) together define the term *SubmittedThesis* as the class of all thesis submitted to somela university.

$$\boxed{\begin{array}{lrl}
(1) & \exists worksOn.\{CompSci\} & \sqsubseteq CompScientist \\
(2) & PhD & \sqsubseteq \exists hasAdvisor.PhD \\
(3) & PhD & \sqsubseteq \exists wroteThesis.SubmittedThesis \\
(4) & \exists wroteThesis.SubmittedThesis & \sqsubseteq PhD \\
(5) & Thesis \sqcap \exists submittedTo.University & \sqsubseteq SubmittedThesis \\
(6) & SubmittedThesis & \sqsubseteq Thesis \sqcap \exists submittedTo.University
\end{array}}$$

Fig. 2: An example MGP ontology

[1] http://genealogy.math.ndsu.nodak.edu/

Now consider the following conjunctive queries:

$$\begin{aligned} q_1(x,y) =& \mathit{ComputerScientist}(x), \mathit{ComputerScientist}(y), \mathit{hasAdvisor}(x,y),\\ & \mathit{wroteThesis}(x,z), \mathit{wroteThesis}(y,z'),\\ & \mathit{submittedTo}(z,w), \mathit{submittedTo}(z',w)\\ q_2(x) =& \mathit{ComputerScientist}(x), \mathit{hasAdvisor}(x,y),\\ & \mathit{wroteThesis}(y,z), \mathit{submittedTo}(z,w), \mathit{University}(z',w) \end{aligned}$$

The query q_1 asks for pairs (x, y) of computer scientists that submitted their thesis to the same university w. The pair $(\mathit{JeffreyUllman}, \mathit{AlbertoMendelzon})$ is one such pair. They both submitted their thesis to *Princeton* and they are both instances of *ComputerScientist*. Note that the data does not explicitely state the latter, but it is implied by axiom (1) and the fact that they both work on the *CompScience* field.

The query q_2 asks for computer scientists whose advisor submitted his thesis to a university. The answer includes both *AlbertoMendelzon* and *JeffreyUllman*. Even though the data contains nothing about the PhD advisor of *JeffreyUllman*, the ontology implies that he must have one, who is a PhD holder and hence must have submitted a thesis to some university.

In the absence of an ontology, OBQA coincides with standard query evaluation over plain (graph) databases, but in general, the presence of the ontology makes query answering more involved. As we have seen, to answer a query one may need to take into account instances of concepts and roles not given in the data, like for the concept *CompScientist* in query q_1. More interestingly, we may even need to consider variable assignments that map query variables to unknown objects that are not named constants in the database: to obtain *JeffreyUllman* as an answer to q_2, we must map variables y, z, and w to objects we known nothing about, but whose existence is implied by the axioms.

As we will see in the next section, the actual impact of the ontology on (the computational complexity of) the query answering problem depends on the specific description logic in which the ontology is written, and on the kind of query that is to be answered. In this paper, we briefly discuss some of the main challenges to be tackled when solving the OBQA problem, for different DLs and query languages, and survey the complexity results obtained so far.

The rest of the paper is organized as follows. In the next section, we give some DL and OBQA preliminaries. They are rather technical and given for the sake of completeness, but it is not necessary to read them in detail for understanding the rest of the material. Finally, section 3 is the core of the paper, where we discuss the challenges that arise in the OBQA setting and complexity results. The interested reader may refer to [21] for a more detailed survey of OBQA.

2 Preliminaries

We briefly recall of some popular DLs. We start giving the syntax and semantics of concepts and roles, and then describe how they can be used to write ABoxes and ontologies in different DLs.

A DL vocabulary consists of three infinite, countable, disjoint sets $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ of concept names, role names, and individuals. Different DLs provide different *concept and role constructors*, which can be used to combine the terms in the vocabulary into complex *concepts* and *roles*. The only role constructor we consider here is the *inverse*: for $r \in \mathsf{N_R}$, its inverse r^- is also a role. We use $\overline{\mathsf{N_R}}$ to refer to the set $\mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$ of all roles, and if $R \in \overline{\mathsf{N_R}}$, we use R^- to mean r^- if $R = r$ and r if $R = r^-$. The most popular concept constructors are summarized in Table 1.

Constructor	Syntax	Semantics
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap C'$	$C^{\mathcal{I}} \cap C'^{\mathcal{I}}$
disjunction	$C \sqcup C'$	$C^{\mathcal{I}} \cup C'^{\mathcal{I}}$
universal restriction	$\forall R.C$	$\{c \mid \forall d, (c,d) \in R^{\mathcal{I}} \to d \in C^{\mathcal{I}}\}$
existential restriction	$\exists R.C$	$\{c \mid \exists d.(c,d) \in R^{\mathcal{I}} \land d \in C^{\mathcal{I}}\}$
nominal	$\{a\}$	$a^{\mathcal{I}}$
qualified number	$\geqslant n\, R.C$	$\{c \mid \mathit{card}(\{d \mid (c,d) \in R^{\mathcal{I}} \land d \in C^{\mathcal{I}}\}) \geq n\}$
restrictions	$\leqslant n\, R.C$	$\{c \mid \mathit{card}(\{d \mid (c,d) \in R^{\mathcal{I}} \land d \in C^{\mathcal{I}}\}) \leq n\}$

Table 1: Summary of most popular DL concept constructors. Here C, C' are concepts, R is a role, $a \in \mathsf{N_I}$ an individual, and $n \geq 0$ a non-negative integer.

Semantics. As usual, the semantics of DLs is defined in terms of interpretations of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the *domain* $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is an *interpretation function* mapping each $a \in \mathsf{N_I}$ to an object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each $A \in \mathsf{N_C}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is inductively extended to general concepts and roles. For inverse roles, we have $(r^-)^{\mathcal{I}} = \{(d,c) \mid (c,d) \in r^{\mathcal{I}}\}$. The semantics of the concept constructors in Table 1 is summarized in the last column.

2.1 ABoxes and Ontologies

A DL *knowledge base* consists of an *ABox*, and a *TBox* or *ontology*.[2] In all the DLs we consider, an *ABox* is defined a set of *assertions* which may take the form $A(b)$ or $r(a,b)$ with $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$.

[2] Sometimes the term *ontology* is used to refer to a knowledge base. Here we use it as a synonym of TBox.

The ontology is a set of axioms, whose form depends on the DL in question. In this paper we mention the following DLs:

The DL-Lite family [7, 1] is a prominent family of *lightweight DLs*, specially tailored in such a way that they can describe complex conceptual models, yet the complexity of reasoning is low and query answering can be achieved by efficient and scalable algorithms. DL-Lite is the most popular family of DLs for OBDA, and it underlies the OWL 2 QL profile of the OWL standard. Many dialects of DL-Lite are known, but we focus on the following three:

- The core dialect $\textit{DL-Lite}_{core}$.
- $\textit{DL-Lite}_{\mathcal{R}}$, the extension of $\textit{DL-Lite}_{core}$ with role inclusions of the form $R \sqsubseteq S$ and $R \sqsubseteq \neg S$, which is very closely related to OWL 2 QL.
- $\textit{DL-Lite}_{\mathsf{RDFS}}$, a fragment of $\textit{DL-Lite}_{\mathcal{R}}$ that is probably less known, but quite interesting for comparison purposes. It disables negation and existential quantification in the right hand side of axioms, and it is roughly equivalent to RDF(S), the schema language for RDF [14].

The interested reader can find the full syntax of these languages in the first block of Table 2. As usual for DL-Lite, we use *unqualified* existentials and write $\exists R$ rather than $\exists R.\top$, where $\top$ is a shortcut for $A \sqcup \neg A$.

The $\mathcal{EL}$ family [2] is the other prominent family of lightweight DLs, and it underlies the OWL 2 EL profile. The $\mathcal{EL}$ family is particularly popular for life science ontologies. As we see below, like $\textit{DL-Lite}_{core}$, the $\mathcal{EL}$ family has limited expressive power but allows for efficient reasoning. In Table 2, we recall the syntax of two DLs in this family: the basic $\mathcal{EL}$, and $\mathcal{ELH}$, its extension with role inclusions $r \sqsubseteq s$.

Expressive DLs that fully support Boolean concept constructors and both kinds of quantifiers have traditionally been the main focus of attention of the DL community. Here we recall the well-known languages $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$, which roughly correspond the the OWL Lite and OWL DL profiles of the first generation of OWL standards. Their syntax is defined in Table 2; there is an additional syntactic requirement that disallows *non-simple roles* (i.e., roles that have a transitive subrole) from occurring in number restrictions. The full OWL 2 is based on a logic called $\mathcal{SROIQ}$, but we do not include it in the table since its syntax is quite cumbersome to define.

Horn DLs that are obtained from expressive DLs by fully disallowing disjunction. The syntax of Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SHOIQ}$, in normal form [16], is given in the last block of the table.

Semantics of ABoxes and Ontologies An interpretation $\mathcal{I}$ satisfies an assertion $A(a)$ (resp. $r(a, b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$), and $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies all assertions in $\mathcal{A}$. For the TBox axioms, we have that $\mathcal{I}$ satisfies an inclusion $G \sqsubseteq H$ (where G and H are either two concepts, or two roles) if $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$; it satisfies $\mathsf{trans}(R)$ if $R^{\mathcal{I}}$ is a transitive relation. $\mathcal{I}$ is a model of an ontology $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$, and $\mathcal{I}$ is a *model* of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I}$ satisfies all inclusions in $\mathcal{T}$ and all assertions in $\mathcal{A}$.

DL	TBox axioms
$DL\text{-}Lite_{core}$	$B \sqsubseteq C$ $\quad B ::= A \mid \exists R \quad C ::= B \mid \neg B \quad R ::= r \mid r^-$
$DL\text{-}Lite_{\mathcal{R}}$	$B \sqsubseteq C,\ R \sqsubseteq S$ $\quad B ::= A \mid \exists R \quad C ::= B \mid \neg B \quad R ::= r \mid r^- \quad S ::= R \mid \neg R$
$DL\text{-}Lite_{\mathsf{RDFS}}$	$B \sqsubseteq A,\ R \sqsubseteq S$ $\quad B ::= A \mid \exists R \quad R, S ::= r \mid r^-$
$\mathcal{EL}$	$C \sqsubseteq D$ $\quad C, D ::= A \mid C \sqcap D \mid \exists r.C$
$\mathcal{ELH}$	$C \sqsubseteq D, r \sqsubseteq s$ $\quad C, D ::= A \mid C \sqcap D \mid \exists r.C$
$\mathcal{SHIQ}$	$C \sqsubseteq D, R \sqsubseteq S$ $\mathsf{trans}(R)$ $\quad C, D ::= A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \geqslant n\, R.C \mid \leqslant n\, R.C$ $\quad R, S ::= r \mid r^-$
$\mathcal{SHOIQ}$	$C \sqsubseteq D, R \sqsubseteq S$ $\mathsf{trans}(R)$ $\quad C, D ::= A \mid \{a\} \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \geqslant n\, R.C \mid \leqslant n\, R.C$ $\quad R, S ::= r \mid r^-$
Horn-$\mathcal{SHIQ}$	$B \sqcap B' \sqsubseteq C,\ \exists R.B \sqsubseteq C,\ C \sqsubseteq \forall R.B,\ C \sqsubseteq \exists R.B$ $\quad B, B', C ::= A$ $\quad C \sqsubseteq\ \geqslant n\, R.B,\ C \sqsubseteq\ \leqslant 1\, R.B,\ R \sqsubseteq S,\ \mathsf{trans}(R)$ $\quad R, S ::= r \mid r^-$
Horn-$\mathcal{SHOIQ}$	$B \sqcap B' \sqsubseteq C,\ \exists R.B \sqsubseteq C,\ C \sqsubseteq \forall R.B,\ C \sqsubseteq \exists R.B$ $\quad B, B', C ::= A \mid \{a\}$ $\quad C \sqsubseteq\ \geqslant n\, R.B,\ C \sqsubseteq\ \leqslant 1\, R.B,\ R \sqsubseteq S,\ \mathsf{trans}(R)$ $\quad R, S ::= r \mid r^-$

Table 2: Summary of some popular DLs. Here $A \in \mathsf{N_C}$ is a concept name, $r, s \in \mathsf{N_R}$ are role names, a is an individual and $n \geq 0$ is a non-negative integer. Special restrictions apply to roles in number restrictions in (Horn)-$\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$.

2.2 Query Languages

Most work on OBQA and OBDA so far has considered *conjunctive queries* and their unions as query languages. We also define *instance queries*, which are CQs with only one atom.

Definition 1 (IQs,CQs,UCQs). *Let* $\mathsf{N_V}$ *denote a countably infinite set of* variables*. A* conjunctive query (CQ) *(with answer variables* $\boldsymbol{x}$*) is an expression* $q(\boldsymbol{x}) = \exists \boldsymbol{y}.\varphi$*, where* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are tuples of variables from* $\mathsf{N_V}$*, and* φ *is a conjunction of atoms of the forms*

(i) $A(t)$*, where* $A \in \mathsf{N_C}$ *is a concept name and* $t \in \mathsf{N_I} \cup \boldsymbol{x} \cup \boldsymbol{y}$*, and*
(ii) $r(t, t')$*, where* $r \in \mathsf{N_R}$ *is a role name and* $t, t' \in \mathsf{N_I} \cup \boldsymbol{x} \cup \boldsymbol{y}$*.*

A query with $\boldsymbol{x} = \langle\rangle$*, that is, where all variables in* φ *are existentially quantified, is called* Boolean*. If in a CQ* $q(\boldsymbol{x}) = \exists \boldsymbol{y}.\varphi$ *we have that* φ *consists of only one atom, we call it an* instance query (IQ)*.*

A union of conjunctive queries (UCQ) is a disjunction of CQs with the same answer variables, that is, an expression $q(\boldsymbol{x}) = \exists \boldsymbol{y_1}\varphi_1 \vee \cdots \vee \exists \boldsymbol{y_n}\varphi_n$ *where each* φ_i *is a conjunction of atoms as above, with arguments from* $\mathsf{N_I} \cup \boldsymbol{x} \cup \boldsymbol{y_i}$*.*

One of the fundamental differences between query answering with DL ontologies in contrast to plain graph databases, is that in the former setting researchers have focused on CQs and UCQs, while for the latter setting the basic querying mechanism are regular path queries (RPQs) and their extensions. These queries enable the sophisticated navigation of paths that has long been considered crucial for querying data on the web. Recent work has aimed at bridging this gap, by developing OBQA algorithms for answering different forms of RPQs [5, 4], like the ones we define next.

Definition 2 ((C)(2)RPQs). *A* conjunctive two-way regular path query (C2RPQ) *(with answer variables $\boldsymbol{x}$) has the form $q(\boldsymbol{x}) = \exists \boldsymbol{y}.\varphi$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are tuples of variables from $\mathsf{N_V}$, and φ is a conjunction of atoms of the form (i) above, and of the following form:*

(ii') $\rho(t,t')$, where ρ is a regular expression over the alphabet $\overline{\mathsf{N_R}} \cup \{A? \mid A \in \mathsf{N_C}\}$, and $t,t' \in \mathsf{N_I} \cup \boldsymbol{x} \cup \boldsymbol{y}$.

Conjunctive (one-way) regular path queries (CRPQs) *are obtained by disallowing inverse roles from $\overline{\mathsf{N_R}} \setminus \mathsf{N_R}$ in atoms of type (ii).* Two-way regular path queries (2RPQs) *consist of a single atom of type (ii), and* regular path queries (RPQs) *further disallow inverse roles.*

2.3 Query Answering

We now define formally the OBQA problem we discuss. In what follows, we use the term *query* to refer to a query of any of the kinds defined above.

Definition 3 (Query match, (certain) answers). *Let $\mathcal{I}$ be an interpretation. A match for a query $q(\boldsymbol{x}) = \exists \boldsymbol{y}.\varphi$ in $\mathcal{I}$ is a mapping π form the terms occurring in q to the domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ such that $\pi(a) = a^{\mathcal{I}}$ for every individual a occurring in q, and that satisfies:*

(i) $\pi(t) \in A^{\mathcal{I}}$ for every atom $A(t)$ in φ,
(ii) $(\pi(t), \pi(t')) \in r^{\mathcal{I}}$ for every atom $r(t,t')$ in φ, and
*(ii') $\pi(t')$ is a ρ-successor of $\pi(t)$ for every atom $\rho(t,t')$ in φ; where an element $d \in \Delta^{\mathcal{I}}$ is called a ρ-*successor *of $c \in \Delta^{\mathcal{I}}$ in $\mathcal{I}$ if there is some chain $c_1, \ldots, c_n$ of objects from $\Delta^{\mathcal{I}}$ and a chain $R_1, \ldots, R_{n-1}$ of roles in the language of ρ such that $c_1 = c$, $c_n = d$, and $(c_i, c_{i+1}) \in R_i^{\mathcal{I}}$ for $1 \leq i < n$.*

Let $\boldsymbol{x} = x_1, \ldots, x_m$. The answers to q in $\mathcal{I}$ are the tuples $a_1, \ldots, a_m$ such that there is a match π for q in $\mathcal{I}$ with $\pi(x_i) = a_i$ for $1 \leq i \leq m$.
Finally, let $\mathcal{A}$ be an ABox and $\mathcal{T}$ an ontology. The (certain) answers *to q over $\mathcal{I}$ are the tuples of individuals that are an answer to q in every model $\mathcal{I}$ of $(\mathcal{A}, \mathcal{T})$.*

We are interested in the decision problem associated to query answering, sometimes called *query output tuple (QOT) problem*, which consists on deciding whether a given tuple of individuals is a certain answer to a given query. For Boolean queries, whose only possible answer is the empty tuple, one usually talks about *query entailment*. Deciding query entailment amounts to deciding whether there exists a query match in every model of the ABox and the ontology. It is well known that the query output tuple problem for an arbitrary query reduces linearly to the entailment problem of a Boolean query. We summarize the main OBQA problems that we discuss in this paper in Table 3.

Problem	Varying input	Fixed input	Problem definition
Data complexity			
QueryAnswer	$\mathcal{A}$	q, $\mathcal{T}$, $\boldsymbol{a}$	Is $\boldsymbol{a}$ in the answers of q over $(\mathcal{T}, \mathcal{A})$?
QueryEntailment	$\mathcal{A}$	q, $\mathcal{T}$	Is there a match for q in every model of $(\mathcal{T}, \mathcal{A})$?
Combined complexity			
QueryAnswer	q, $\mathcal{T}$, $\mathcal{A}$, $\boldsymbol{a}$		Is $\boldsymbol{a}$ in the answers of q over $(\mathcal{T}, \mathcal{A})$?
QueryEntailment	q', $\mathcal{T}$, $\mathcal{A}$		Is there a match for q in every model of $(\mathcal{T}, \mathcal{A})$?

Table 3: Main OBQA problems. Here $\mathcal{T}$ is an ontology, $\mathcal{A}$ an ABox, $\boldsymbol{a}$ a tuple of individuals, q an arbitrary query, and q' a Boolean query.

Data and combined complexity We are interested in two measures of complexity. First, since the size of the data repositories usually dominates over the size of the ontology and the query, we are interested in measuring the complexity in terms of the ABox only. This measure of complexity is called *data complexity*, and like for traditional databases, for OBQA and OBDA it is considered the most relevant in practice. The other notion is called *combined complexity* and it takes all components as an input: the query, the ontology, the ABox, and possibly the tuple of constants.

3 The challenges of OBQA

Recall that in OBQA we view the ABox as a database that may be incomplete in at least two ways:

1. The existing objects may be instances of concepts and roles that do not show up in the data.
2. The ontology may imply the existence of more objects, beyond the named individuals that explicitely appear in the data. While these objects do not participate in query answers, they do participate in query matches.

A naïve way to approach the problem would be to complete the data as prescribed by the ontology, and then evaluate the query over the resulting structure. However, this strategy may not work. We are interested in the *certain answers* over all models, that is, over all ways to complete the data to satisfy the ontology. But there may be too many ways to do this completion, and the completed structures (models) may be too large, and are often infinite. We discuss below how these two problems make OBQA, in general, a computationally harder problem that query answering over plain databases, as Table 4 shows. We remark that, for all DLs, the complexity of IQs coincides with the complexity of standard reasoning tasks (like subsumption and satisfiability).

Before discussing the results in the table in more detail, we want to point out that most DLs are fragments of first order logic (FOL), and an ontology can

	Combined complexity		**Data complexity**	
	IQs	CQs	IQs	CQs
Plain databases	in AC$_0$	NP-c	in AC$_0$	in AC$_0$
DL-Lite$_{(\mathcal{R})}$	NL-c	NP-c	in AC$_0$	in AC$_0$
$\mathcal{EL}$, $\mathcal{ELH}$	P-c	NP-c	P-c	P-c
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases *DL-Lite*$_{\text{RDFS}}$	NL-c	NP-c	NL-c	NL-c
DL-Lite$_{(\mathcal{R})}$	P-c†	PSpace-c	NL-c	NL-c
$\mathcal{EL}$, $\mathcal{ELH}$	P-c	PSpace-c	P-c	P-c
	IQs,(2)RPQs	CQs,C(2)RPQs	IQs,(2)RPQs	CQs, C(2)RPQs
Horn-$\mathcal{SHIQ}$, Horn-$\mathcal{SHOIQ}$	ExpTime-c	ExpTime-c	P-c	P-c
$\mathcal{SHIQ}$	ExpTime-c	2ExpTime-c	coNP-c	coNP-c
$\mathcal{SHOIQ}$	co-NExpTime-c	open	coNP-c	open

Table 4: The complexity of QueryAnswer and QueryEntailment. The 'c' indicates completeness results. †P-hardness for RPQs applies only to *DL-Lite*$_{\mathcal{R}}$. For references, see [21, 5].

be rewritten as a FOL formula in a straightforward way [3]. CQs and UCQs are also special cases of FOL formulas, hence the (U)CQ query answering and query entailment problems are special cases of logical consequence in FOL. Even when the query (or the DL, in cases not discussed here) has regular expressions, there is a natural way to view the problem as logical consequence in some extension of FOL that supports transitive closure. However, there is no natural fragment that is decidable and that allows to express both the ontology and the query, hence we can not inherit decidability or complexity results easily.

3.1 Multiple Models

Query answering algorithms from databases usually evaluate the query over one single input structure, the database. In the presence of DL ontologies, however, the query has to be evaluated over all the structures that are models of the data and the ontology. A knowledge base $(\mathcal{T}, \mathcal{A})$ has, in principle, infinitely many models. A trivial reason for this is that we can use arbitrary sets as domains in interpretations, and we can add additional, possibly irrelevant information to any model in a way that it is still a model. But even if we disregard these trivial reasons and consider only the 'meaningful' differences between models, resulting from different ways in which the data may be completed to satisfy all constraints given by the ontology, we can still end up with a very large number of models.

To develop query answering techniques it is usually possible to show that a finite subset of all the models suffices. In the case of Horn DLs, and in particular for the lightweight DLs of the *DL-Lite* and $\mathcal{EL}$ families, we can go even further and show the following key property:

Canonical model property: given a satisfiable knowledge base $(\mathcal{T}, \mathcal{A})$, one can construct a *canonical model* $\mathcal{I}_{(\mathcal{T},\mathcal{A})}$ such that, for every query q, the answers to q over $(\mathcal{T}, \mathcal{A})$ coincide with the answers to q in $\mathcal{I}_{(\mathcal{T},\mathcal{A})}$.

That is, the canonical model $\mathcal{I}_{(\mathcal{T},\mathcal{A})}$ suffices for answering all queries. This property plays a central role in the data complexity of query answering. In fact, all tractability results in the Table 4 (first two blocks and first row of the last block) are for logics that have the canonical model property.

In contrast, the presence of any kind of disjunction in the ontology results in the existence of models that differ in the queries they entail, and makes the query entailment problem coNP hard in data complexity, even for the simplest instance queries. This applies to $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$, which support full Booleans (see the last two rows of Table 4). It also applies to all other expressive DLs, like $\mathcal{SROIQ}$ which underlies OWL 2. The need to consider several models also has a big impact on the combined complexity of CQ answering: it makes the problem exponentially harder than standard reasoning for $\mathcal{SHIQ}$ and most expressive DLs (for $\mathcal{SHOIQ}$, decidability of the problem remains open).

3.2 Large and Infinite Models

Unfortunately, identifying one model, or one set of models, that is sufficient for deciding query entailment, is not enough to make query answering easy. Usually we must overcome an additional difficulty: (the domain of) models can be, in general, infinite. This is caused by existential concepts in the right hand side of axioms, a feature that is considered fundamental in DLs and allowed in all the logics we have mentioned, except for *DL-Lite*$_{\mathsf{RDFS}}$.

Many DLs enjoy the *finite model property.* That is, every satisfiable KB has a model where the domain is a finite set. For lightweight DLs like *DL-Lite* and $\mathcal{EL}$, there is even a model where the domain's cardinality is polynomially bounded in the size of the KB. However, these positive properties are of limited use for query answering. Small and finite models are usually built by 'reusing' domain elements to satisfy existential restrictions. This creates cycles in models that may not affect KB satisfiability, but often result in spurious query matches that are not really implied by the KB. For example, one can avoid generating an infinite R-chain of objects for satisfying an axiom $A \sqsubseteq \exists R.A$ by simply 'reusing' one element e and making it an R-successor of itself. However, this would cause the query $\exists x.R(x, x)$ to be erroneously entailed.

Researchers have come up with a number of different query answering techniques, which overcome the challenges above in different ways. We briefly discuss some of them below, and refer to [21] for a more detailed discussion. In general, the results obtained so far suggest that the size of models is easier to tame than their number. For DLs that enjoy the canonical model property, even if canonical models are infinite, several query answering algorithms have been proposed and implemented. Intuitively, algorithms can build on the fact that canonical models, even if they are infinite, enjoy a relatively regular structure and can be compactly represented. In contrast, for expressive DLs like $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$

the need to consider multiple models for query answering, and the interaction of these models with the query, results in much more complex algorithms that have eluded implementation until now.

3.3 Query Answering Techniques

Most work on OBQA focuses on lightweight DLs and uses *rewriting techniques*. The idea here is to get rid of the ontological knowledge in a first step, and then use existing database technologies. That is, one reduces query answering in the presence of an ontology to answering another query, possibly in a different language, but over a single relational structure (a plain database).

Query Rewriting for *DL-Lite* The rewriting approach was first proposed for *DL-Lite*, since this family of logics enjoys a remarkable property: given an ontology $\mathcal{T}$ and a CQ q, we can compile all the knowledge from $\mathcal{T}$ that is relevant for answering q into q itself, to obtain another query (a UCQ) that can be valuated over the data only.

> *FOL Rewritability:* Given a *DL-Lite* ontology $\mathcal{T}$ and a CQ q, one can compute a FOL query $q^{\mathcal{T}}$ such that, for every ABox $\mathcal{A}$, the answers of q over $(\mathcal{A}, \mathcal{T})$ coincides with the answers of $q^{\mathcal{T}}$ over $\mathcal{A}$.

This means we can reduce the problem of answering a CQ over a *DL-Lite* knowledge base to simply evaluating a FOL query, or equivalently, an SQL query, over a standard database, a well understood problem for which efficient solutions are readily available. Since the rewriting of q into $q^{\mathcal{T}}$ does not depend on the ABox $\mathcal{A}$, this also implies that the data complexity of CQ answering over *DL-Lite* KBs is not higher than the data complexity of evaluating FOL queries over standard databases, namely in the class AC_0 (see Table 4, first and second rows).

FOL Rewritability was first exploited to develop OBQA algorithms for *DL-Lite* in [7], where the authors proposed the *perfect reformulation* (PerfectRef) algorithm for reformulating a CQ q into a UCQ $q^{\mathcal{T}}$. Intuitively, the algorithm uses $\mathcal{T}$ to replace concepts and roles in the query by concepts and roles that imply them. For example, in the presence of an ontology containing axioms $B \sqsubseteq C, \exists R \sqsubseteq B$, the query $q = A(x), C(y)$ can be rewritten into queries $q' = A(x), B(y)$ and $q''_2 = A(x), R(y, z)$, where z is a fresh, unbound variable. Different choices for replacing a given concept or role result in different queries, and the exhaustive application of the rewriting rules generates a set of CQs, that is, a UCQ. To obtain a complete query that covers all possible ways of implying the concept and roles in the input q, it is necessary to allow the unification of query variables in different atoms. The algorithm shows that the problem is in NP in combined complexity, which is not higher than for plain databases. Many later works have focused on improving the PerfectRef algorithm, for example by optimizing the unification step or rewriting into a more succinct language than UCQs, e.g. into non-disjunctive Datalog [23].

Query rewriting beyond *DL-Lite* It is well known that CQ answering in practically all DLs outside the *DL-Lite* family is at least LogSpace hard in data complexity, and hence FO rewritability is the lost. This applies, in particular, to the $\mathcal{EL}$ family and to all its extensions, like Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SHOIQ}$. However, this kind of logics still have the canonical model property, and other rewriting techniques have been proposed that rewrite the query and the ontology into more expressive languages, like Datalog.

One such approach, developed for Horn-$\mathcal{SHIQ}$ in [11], removes query variables that can be mapped to unknown elements implied by the existential axioms, and adds to the query concept atoms that imply the existence of these elements, in every possible way. For example, in the presence of an ontology containing an axiom $B \sqsubseteq \exists R.C$, the query $q = A(x), R(x, y), C(y)$ can be rewritten into $q' = A(x), B(x)$. The exhaustive application of this kind of rewriting results in a query that has a match where no unknown objects occur, and can in turn be evaluated over the named individuals only. More specifically, it is evaluated ABox using an additional set of Datalog rules to imply the concepts and roles that each individual is an instance of. This approach yields a tight ExpTime upper bound in combined complexity for Horn-$\mathcal{SHIQ}$, which is already ExpTime-hard for instance queries and standard reasoning. In data complexity, it is polynomial.

A similar approach had already been proposed for $\mathcal{EL}$ in [22], yielding tight upper bounds of NP in combined and P in data complexity. Finally, the *combined approach*, introduced in [18] is another interesting approach for $\mathcal{EL}$, that yields the same complexity bounds. It uses *data completion* to compile the TBox into the *data* rather than into the query. After some polynomial rewritings, the query can be posed over a completed ABox that is polynomially larger than the original one, using standard relational database technologies.

Answering RPQs and their extensions in lightweight DLs As we have mentioned, until recently most work on OBQA focused on CQs and UCQs. RPQs and their extensions had only been explored for very expressive DLs, where they have the same complexity as CQs (see Table 4), and can be answered using techniques like the automata one described below [9, 10]. For lightweight DLs they have only been studied recently. It is easy to see that since RPQs can express reachability, the data complexity jumps to NL-complete even for plain graph DBs, FO rewritability is lost for *DL-Lite*. In combined complexity, the problem is NL-hard for (2)RPQs and NP-hard for C(2)RPQs, already for graph databases. As soon as existential quantification is allowed on the right hand side of axioms and unknown objects may participate in query matches, these bounds increase to P-hard and PSpace-hard respectively. Matching upper bounds were obtained in [5] with a non-trivial adaptation of the rewriting technique for $\mathcal{SHIQ}$ mentioned above.

Techniques for expressive DLs Expressive DLs lack the canonical model property. All algorithms developed so far focus on deciding the existence of a countermodel, that is, a model where the query has no match. They also build on the fact that one can restrict the search to interpretations that have a regular, forest like structure. Techniques to find such a counterexample include automata theoretic ones (which in fact work for C2RPQs) [9, 10], the so-called *rolling-up* [8, 15, 12, 13, 19], and modified tableaux [17, 20]. Some of them are surveyed in [21].

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005) (2005)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
4. Bienvenu, M., Ortiz, M., Simkus, M.: Answering expressive path queries over lightweight dl knowledge bases. In: Kazakov, Y., Lembo, D., Wolter, F. (eds.) Proc. of DL 2012. CEUR Workshop Proceedings, vol. 846. CEUR-WS.org (2012)
5. Bienvenu, M., Ortiz, M., Simkus, M.: Conjunctive regular path queries in lightweight description logics. In: Proc. of IJCAI 2013 (2013), to appear
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: Ontology-based database access. In: Proc. of the 15th Italian Conf. on Database Systems (SEBD 2007). pp. 324–331 (2007)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39(3), 385–429 (2007)
8. Calvanese, D., De Giacomo, G., Lenzerini, M.: Conjunctive query containment and answering under description logics constraints. ACM Trans. on Computational Logic 9(3), 22.1–22.31 (2008)
9. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proc. of the 22nd AAAI Conference on Artificial Intelligence (AAAI 2007). pp. 391–396 (2007)
10. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Boutilier, C. (ed.) Proc. of IJCAI 2009. pp. 714–720 (2009)
11. Eiter, T., Ortiz, M., Šimkus, M., Tran, T.K., , Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012) (2012), to appear
12. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic $\mathcal{SHIQ}$. Journal of Artificial Intelligence Research 31, 151–198 (2008)
13. Glimm, B., Horrocks, I., Sattler, U.: Unions of conjunctive queries in SHOQ. In: Proc. of the 11th International Conference on the Principles of Knowledge Representation and Reasoning (KR-08). pp. 252–262. AAAI Press/The MIT Press (2008)
14. Grau, B.C.: A possible simplification of the semantic web architecture. In: Proceedings of the 13th international conference on World Wide Web. pp. 704–713. WWW '04, ACM, New York, NY, USA (2004), `http://doi.acm.org/10.1145/988672.988769`

15. Horrocks, I., Tessaris, S.: A conjunctive query language for description logic ABoxes. In: Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000). pp. 399–404 (2000)
16. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for Horn description logics. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07). pp. 452–457. AAAI Press (2007)
17. Levy, A.Y., Rousset, M.C.: Combining Horn rules and description logics in CARIN. Artificial Intelligence 104(1–2), 165–209 (1998)
18. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: Proc. of IJCAI 2009. pp. 2070–2075. AAAI Press (2009)
19. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings. Lecture Notes in Computer Science, vol. 5195, pp. 179–193. Springer (2008)
20. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. J. of Automated Reasoning 41(1), 61–98 (2008)
21. Ortiz, M., Simkus, M.: Reasoning and query answering in Description Logics. In: Eiter, T., Krennwallner, T. (eds.) Reasoning Web. Semantic Technologies for Advanced Query Answering - 8th International Summer School 2012, Proceedings. Lecture Notes in Computer Science, vol. 7487, pp. 1–53. Springer (2012)
22. Rosati, R.: On conjunctive query answering in $\mathcal{EL}$. In: Proc. of the 2007 Description Logic Workshop (DL 2007). CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-250/`, vol. 250 (2007)
23. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning (KR 2010) (2010)