Query Answering over Description Logic Ontologies

Diego Calvanese

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano
calvanese@inf.unibz.it

Abstract. Description Logics (DLs) provide the formal foundation for ontology languages, and they have been advocated as formalisms for modeling the domain of interest in various settings, including the Semantic Web, data and information integration, and ontology-based data access. An important requirement there is the ability to answer complex database-like queries, while taking into account both extensional and intensional domain knowledge. The task of answering queries has been investigated intensively in the last years for a variety of DLs, and considering both data complexity, i.e., the complexity measured in the size of the extensional information only, and combined complexity. On the one hand, it has been shown to be in general (exponentially) more difficult than the standard reasoning tasks of concept satisfiability and subsumption; on the other hand a broad range of techniques have been developed. We overview here some of the key techniques developed in the last years for query answering over DL ontologies, ranging from rewriting based approaches for lightweight DLs, to tableaux algorithms, and techniques based on automata on infinite trees for very expressive DLs. The associated results, accompanied by matching lower bounds, have contributed to shaping the computational complexity picture for ontology-based query answering.

1 Introduction

Description Logics [4] (DLs) are a class of logics that are particularly well suited for representing structured knowledge. They have been developed starting from the early 1980s in order to formalize early days knowledge representation formalisms, such as Semantic Networks and Frames, which lacked not only well understood computational properties, but despite their name even a formal semantics. Since then DLs have evolved into a large collection of variants that, in their various forms, subsume essentially all class-based representation formalisms used in Databases, Artificial Intelligence, and Software Engineering. DLs follow the quite common approach used in knowledge representation of modeling the domain of interest in terms of *concepts*, which denote sets of objects, and relations between objects belonging to certain concepts. We consider here "traditional" DLs, which are equipped only with binary relations, called *roles*, but many variants of DLs allowing for the use of relations of arbitrary arity have also been considered in the literature [47,15,14]. Starting from atomic concepts and roles, complex expressions can be constructed inductively by means of suitable concept and role forming operators. Such expressions are then used in an *ontology* to assert knowledge about the domain, both at the intensional level, in the *TBox* of the ontology, and at the extensional level, in the *ABox* of the ontology. Typically, the TBox consists of a set of inclusion assertions between concepts and between roles, where each such assertion states that the set of instances of one concept/role is included in the set of instance of another concept/role. The ABox instead contains facts about individual domain elements, asserting that some individual belongs to a concept, or that some pair of individuals is related by a role.

A distinguishing feature of DLs is the fact that quantification is restricted by the syntax of the logic (which is variable-free) to be *guarded*. A consequence is that DL concept expressions essentially can only be used to represent properties that are encoded by starting from an instance of the concept and following the roles in a *tree-like* navigation. Formally, this aspect is captured by the fact that DLs (in general) satisfy some form of *tree-model property*: if an ontology is satisfiable, then it admits a model that is constituted by a collection[1] of structures that are essentially[2] tree-shaped (when viewing objects as nodes, and relations between objects as edges connecting them). On the one hand, the tree-model property, which DLs share with modal logics and with many variants of program logics [50], accounts for many of the good computational properties of DLs, which, despite being first-order formalism, admit decidable and in many cases efficient inference. On the other hand, the restriction that is at the basis of the tree-model property brings about an intrinsic limitation in expressive power. This makes it impossible in DLs to express inter-relationships between objects that would correspond to following different navigation paths across the data. E.g., in a DL one could not express a concept denoting those individuals for which the house in which they live is located in the same city as the company in which they work. While this kind of restriction is considered acceptable when encoding knowledge at the intensional level in a TBox, it makes DLs not well-suited as a formalism for expressing queries, where in general one is interested in complex inter-relationships between objects.

For this reason, since the late 1990s, researchers have studied the setting where information needs expressed over knowledge encoded in an ontology are formulated not only in terms of concept or role expressions, but also in terms of more complex *queries* in the style of those used in databases [35,15]. In such a setting, where the presence of an ontology accounts for incomplete information, query answering is a form of logical implication, as opposed to model checking, which can be seen as the logical counterpart of query evaluation. For this reason, answering arbitrary (domain independent) FOL queries (corresponding to the core of SQL) over an ontology turns out to be immediately undecidable. Just consider evaluating the query $q(x) = A(x) \wedge \varphi$ over the ontology with an empty TBox and whose ABox just contains $A(c)$, where φ is an arbitrary FOL formula in which A does not appear; then c is in the answer to the query iff φ is valid.

This motivates why in the context of DLs, restricted classes of queries have been considered, which still allow for expressing sufficiently complex interrelationships between data, but do not incur in the computational problems caused by arbitrary FOL queries. A prominent such class of queries are *conjunctive queries* (CQs) [22], corresponding

[1] The ABox accounts for an arbitrary graph-shaped but finite portion of the model, and each object interpreting an individual of the ABox can be seen as the root of a possibly infinite tree originating from that object.

[2] Depending on the constructs of the DL, we might have to account for multiple role-labeled edges between nodes, or for special edges that point back to the ABox part.

to select-project-join SQL queries, and unions thereof (UCQs). Such queries account for the most common type of queries used in the relational setting, and also have found applications in other settings of incomplete information, such as data integration and data exchange [34,25]. DLs, as opposed to relational databases, allow one to represent knowledge in which the underlying data has a rather loose structure, such as the one encountered nowadays in graph databases. Indeed, in the setting of DLs, it became of interest to consider also more flexible mechanisms for querying such kind of data, as the one offered by variants of *regular path queries* [10,1], which allow one to retrieve pairs of objects connected by a path in the data that matches a regular expression over the set of roles (i.e., edge labels in the graph).

In the following, we discuss the challenges that arise when addressing the problem of answering queries over DL ontologies. Given the large amount of parameters that characterize the problem, and the variety of results that have been obtained in the area, our aim is not to be comprehensive, and we refer to [40,38] for recent overviews of query answering in ontologies. Instead, we aim at illustrating three different types of techniques that have been introduced in the literature, and that aimed at addressing different requirements for the problem of query answering over DL ontologies. After introducing in Section 2 some technical preliminaries on DLs and queries that are necessary for the subsequent development, we first illustrate in Section 3 a technique for efficient query answering in lightweight DLs that lends itself for an efficient implementation. Then, in Section 4 we present an adaptation of tableaux algorithms traditionally adopted for DL inference towards query answering in expressive DLs. The technique allows one to obtain optimal complexity bounds in terms of *data complexity*, i.e., when the complexity of the problem is measured in terms of the size of the ABox only. Finally, in Section 5, we present an approach based on automata on infinite trees, that leads to decidability and optimal complexity results (thought not in data complexity) for DLs and query languages that are among the most expressive ones for which decidability of query answering has been established so far.

2 Description Logic Ontologies and Queries

In DLs, the domain of interest is modeled by means of *concepts*, denoting classes of objects, and *roles* (i.e., binary relationships), denoting binary relations between objects.

Syntax of Description Logics. Arbitrary concepts and roles are obtained starting from atomic ones by applying suitable concept and role forming constructs, where the set of allowed constructs characterizes each specific DL. We introduce here the quite expressive DL $\mathcal{ALCOIQHb}^{self}_{reg}$, abbreviated simply as $\mathcal{DL}$, which is a super-language of the various DLs that we consider in this work[3]. Specifically, $\mathcal{DL}$ is an expressive DL in which concepts and roles are formed according to the following syntax:

$$\begin{array}{rcl}
C, C' & \longrightarrow & A \mid C \sqcap C' \mid \forall R.C \mid \neg C \mid \leqslant n\, S.C \mid \{a\} \mid \exists S.\mathsf{Self} \\
P & \longrightarrow & p \mid p^- \\
S, S' & \longrightarrow & P \mid S \cap S' \mid S \setminus S' \mid S \cup S' \\
R, R' & \longrightarrow & \mathsf{T} \mid S \mid R \cup R' \mid R \circ R' \mid R^* \mid id(C)
\end{array}$$

[3] $\mathcal{DL}$ is equivalent to the DL $\mathcal{ZOIQ}$ [20].

where A denotes an *atomic concept*, p an *atomic role*, S, S' simple roles, C, C' arbitrary *concepts*, and R, R' arbitrary *roles*, and a an individual. $\mathcal{DL}$ is obtained from the basic DL language $\mathcal{AL}$ (attributive language) [4], in which one can express concept intersection $C \sqcap C'$ and value restriction $\forall R.C$, by adding several concept and role forming constructs, each indicated by a letter or a sub/super-script in the name of the DL. Such constructs are negation $\neg C$ of arbitrary concepts (indicated by the letter $\mathcal{C}$ in $\mathcal{ALCOIQHb}^{self}_{reg}$), qualified number restrictions $\leqslant n\, S.C$ (indicated by $\mathcal{Q}$), nominals $\{a\}$ (indicated by $\mathcal{O}$), inverse roles p^- (indicated by $\mathcal{I}$), boolean combinations of atomic and inverse roles (indicated by b), the self-construct (indicated by the superscript ^{self}), and regular expressions over roles (indicated by the subscript $_{reg}$). We can also express the top concept $\top$ as an abbreviation for $A \sqcup \neg A$, for some concept A, the bottom concept $\bot$ as $\neg\top$, union $C_1 \sqcup C_2$ as $\neg(\neg C_1 \sqcap \neg C_2)$, and qualified existential quantification on roles $\exists R.C$ as $\neg\forall R.\neg C$. We observe that, in order to preserve decidability of inference, number restrictions are applied only to *simple* roles, i.e., atomic roles p, their inverses p^-, or boolean combinations thereof (see, e.g., [5,11] for the consequences of using more complex roles in number restrictions).

As in most DLs, a $\mathcal{DL}$ *ontology* is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$, the *TBox*, is a finite set of *intensional assertions*, and $\mathcal{A}$, the *ABox*, is a finite set of *extensional* (or, *membership*) *assertions*. Here we consider TBoxes consisting only of inclusion assertions between concepts and between simple roles[4]. An *inclusion assertion* has the form $C \sqsubseteq C'$, with C and C' arbitrary $\mathcal{DL}$ concepts, or $S \sqsubseteq S'$, with S and S' simple roles. Intuitively, it states that, in every model of the TBox, each instance of the left-hand side expression is also an instance of the right-hand side expression. We note that the letter $\mathcal{H}$ in the name of $\mathcal{ALCOIQHb}^{self}_{reg}$ accounts for the presence in the logic of role inclusions, by means of which one can express role *hierarchies*. The ABox consists of a set of extensional assertions, which are used to make statements about individuals. Each such assertion has the form $A(a)$, $p(a, b)$, $a \approx b$, or $a \not\approx b$, with A and p respectively an atomic concept and an atomic role occurring in $\mathcal{T}$, and a, b individuals.

Semantics of Description Logics. We now turn to the semantics of $\mathcal{DL}$, which is given in terms of interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty interpretation domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$, which assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, to each role R a binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$, and to each individual a an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, in such a way that the conditions specified in Figure 1 are satisfied[5]. Unless stated otherwise, we don't make here the *unique name assumption*, i.e., we allow different individuals to be interpreted as the same domain element.

The semantics of a $\mathcal{DL}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is the set of *models* of $\mathcal{O}$, i.e., the set of interpretations satisfying all assertions in $\mathcal{T}$ and $\mathcal{A}$. It remains to specify when an interpretation satisfies an assertion. An interpretation $\mathcal{I}$ *satisfies* an inclusion assertion $C \sqsubseteq C'$ (resp., $R \sqsubseteq R'$), if $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$ (resp., $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$), a membership assertion

[4] The DLs underlying the Web Ontology Language OWL 2, standardized by the W3C [7], feature additional kinds of TBox assertions, which allow one to state, e.g., the symmetry, reflexivity, or transitivity of a role. We do not consider such kinds of assertions here.

[5] We have used "$\circ$" to denote concatenation of binary relations, and "$*$" to denote the reflexive-transitive closure of a binary relation.

$$\begin{aligned}
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
(C \sqcap C')^{\mathcal{I}} &= C^{\mathcal{I}} \cap C'^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{\, o \mid \forall o'.\, (o,o') \in R^{\mathcal{I}} \to o' \in C^{\mathcal{I}} \,\} \\
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\leqslant n\, R.C)^{\mathcal{I}} &= \{\, o \mid \sharp\{o' \in C^{\mathcal{I}} \mid (o,o') \in R^{\mathcal{I}}\} \leq n \,\} \\
\{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\
\exists S.\mathsf{Self}^{\mathcal{I}} &= \{\, o \mid (o,o) \in S^{\mathcal{I}}\}
\end{aligned}
\qquad
\begin{aligned}
p^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
(p^-)^{\mathcal{I}} &= \{(o,o') \mid (o',o) \in P^{\mathcal{I}}\} \\
(S \sqcap S')^{\mathcal{I}} &= S^{\mathcal{I}} \cap S'^{\mathcal{I}} \\
(S \setminus S')^{\mathcal{I}} &= S^{\mathcal{I}} \setminus S'^{\mathcal{I}} \\
(\mathsf{T})^{\mathcal{I}} &= \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
(R \cup R')^{\mathcal{I}} &= R^{\mathcal{I}} \cup R'^{\mathcal{I}} \\
(R \circ R')^{\mathcal{I}} &= R^{\mathcal{I}} \circ R'^{\mathcal{I}} \\
(R^*)^{\mathcal{I}} &= (R^{\mathcal{I}})^*
\end{aligned}$$

Fig. 1. Interpretation of $\mathcal{DL}$ concepts and roles

$A(a)$ (resp., $p(a,b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp., $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in p^{\mathcal{I}}$), an assertion of the form $a \approx b$ if $a^{\mathcal{I}} = b^{\mathcal{I}}$, and an assertion of the form $a \not\approx b$ if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

The basic reasoning task in $\mathcal{DL}$ is that of *logical implication*, i.e., checking whether a TBox or ABox assertion is implied by an ontology, i.e., holds in every model of the ontology. All other ontology-level inference tasks, such as checking whether an ontology is satisfiable (i.e., admits a model), or whether a concept is satisfiable with respect to an ontology, can easily be reduced to logical implication. Like in many other expressive DLs, reasoning in $\mathcal{DL}$, is decidable in deterministic exponential time, and actually ExpTime-complete, see, e.g., [4].

We consider here various sub-languages of $\mathcal{DL}$. The logics $\mathcal{ALC}[\mathcal{O}][\mathcal{I}][\mathcal{Q}]\mathcal{H}[b_{reg}^{self}]$ are obtained from $\mathcal{DL}$ by possibly dropping some of the $\mathcal{O}$, $\mathcal{I}$, and $\mathcal{Q}$ constructs, or all of boolean combinations, regular expressions over roles, and the self-construct. Moreover, in the lightweight DL *DL-Lite*$_{\mathcal{R}}$ [13], inclusion assertions have one of the forms:

$$B \sqsubseteq B' \qquad B \sqsubseteq \neg B' \qquad\qquad P \sqsubseteq P' \qquad P \sqsubseteq \neg P'$$

Here, roles P, P' are either an atomic role p or the inverse p^- of an atomic role, and *basic concepts* B, B' are constructed according to the following syntax:

$$B, B' \;\longrightarrow\; A \mid \exists P$$

where we use $\exists P$ as an abbreviation for $\exists P.\top$. Intuitively, a basic concept denotes either an atomic concept A, or the projection of a role p on its first component ($\exists p$) or second component ($\exists p^-$). Despite its simplicity, *DL-Lite* is able to capture the essential features of most conceptual modeling formalisms, such as UML Class Diagrams or Entity-Relationship schemata (see, e.g., [12]).

Queries. We introduce now *positive two-way regular path queries* (P2RPQs), which are a quite general class of queries that subsumes most of the query formalisms that have been considered in the context of query answering under incomplete information. A P2RPQ $q(\vec{x})$ has the form $\exists \vec{y}.\varphi$, where $\vec{x}$ and $\vec{y}$ are tuples of variables and φ is a formula built using $\wedge$ and $\vee$ from atoms of the form $C(v)$ and $R(v, v')$, where v, v' are variables from $\vec{x}$, from $\vec{y}$, or individuals, C is a (possibly complex) concept, and R is a (possibly complex) $\mathcal{DL}$ role. We call $\vec{x}$ the *answer* (or *distinguished*) *variables*, and $\vec{y}$

the *existential variables* of q. A Boolean query is one where $\vec{x}$ is the empty tuple $\langle\rangle$, i.e., all variables in the query are existential ones.

A P2RPQ consisting of a single role atom is a *two-way regular path query* (2RPQ), while one consisting of a conjunction of role atoms is a *conjunctive 2RPQ* (C2RPQ). When we do not allow for the use of inverse roles, we obtain the one-way variants of the above query languages, i.e., PRPQs/CRPQs/RPQs. When we further restrict P2RPQs so as to forbid the use of regular expressions in role atoms, we obtain the class of *positive queries* (PQs), and when we forbid also the use of disjunction, we obtain the well known class of *conjunctive queries* (CQs) [22,2]. A *unions of CQs* (UCQs) is a disjunction of CQs with the same answer variables. We observe that the possibility of using regular role expressions in the query atoms of (P/C)(2)RPQs significantly increases the expressive power of the query language, since it allows one to express complex navigations in the models of the given ontology, similar to those possible with (C)(2)RPQs studied in the setting of graph databases [26,16,17,18].

Given a P2RPQ $q(\vec{x}) = \exists\vec{y}.\varphi$ and an interpretation $\mathcal{I}$, let π be a total function from the variables and individuals occurring in q to $\Delta^{\mathcal{I}}$ such that $\pi(a) = a^{\mathcal{I}}$ for each individual a occurring in q. We write $\mathcal{I},\pi \models C(v)$ if $\pi(v) \in C^{\mathcal{I}}$, and $\mathcal{I},\pi \models R(v,v')$ if $(\pi(v),\pi(v')) \in R^{\mathcal{I}}$. Let γ be the Boolean expression obtained from φ by replacing each atom α in φ with $true$, if $\mathcal{I},\pi \models \alpha$, and with *false* otherwise. If γ evaluates to $true$, we say that π is a *match for q in $\mathcal{I}$*, denoted $\mathcal{I},\pi \models q$. When there is some match for q in $\mathcal{I}$, we also say that q can be *mapped to $\mathcal{I}$*.[6] The *answers* to $q(\vec{x})$ in $\mathcal{I}$, is the set $\mathsf{ans}(q,\mathcal{I}) = \{\pi(\vec{x}) \mid \mathcal{I},\pi \models q\}$ of tuples of elements of $\Delta^{\mathcal{I}}$ to which the answer variables of q can be mapped by some match for q in $\mathcal{I}$. Given an ontology $\mathcal{O}$ and a query $q(\vec{x})$, a *certain answer* to q over $\mathcal{O}$ is a tuple $\vec{a}$ of individuals in $\mathcal{O}$ such that $\vec{a}^{\mathcal{I}} \in \mathsf{ans}(q,\mathcal{I})$, for every model $\mathcal{I}$ of $\mathcal{O}$. Note that, while an answer to a query over an interpretation $\mathcal{I}$ is a set of elements of $\Delta^{\mathcal{I}}$, a certain answer is a tuple of individuals appearing in $\mathcal{O}$. We denote with $\mathsf{cert}(q,\mathcal{O})$ the set of certain answers to q over $\mathcal{O}$.

For a Boolean query $q()$, we say that $\mathcal{I}$ *satisfies* $q()$, written $\mathcal{I} \models q()$, if there is some match for $q()$ in $\mathcal{I}$. We have that $\mathsf{cert}(q(),\mathcal{O})$ is either the empty tuple of individuals $\langle\rangle$ (representing $true$), when $\mathcal{I} \models q()$ for every model $\mathcal{I}$ of $\mathcal{O}$, or the empty set $\emptyset$ (representing *false*). In the former case, i.e., when $\mathsf{cert}(q,\mathcal{O}) = \{\langle\rangle\}$, we say that $\mathcal{O}$ *entails* q, denoted $\mathcal{O} \models q$.

Query evaluation consists in computing, given an ontology $\mathcal{O}$ and a P2RPQ $q(\vec{x})$, the set $\mathsf{cert}(q,\mathcal{O})$ of certain answers. The corresponding decision problem is the *recognition problem for query answering*, in which one wants to check whether a given a tuple $\vec{a}$ of individuals is in $\mathsf{cert}(q,\mathcal{O})$. When q is a Boolean query, the corresponding task is *query entailment*, which consists in verifying whether $\mathcal{O} \models q$. In fact, the recognition problem for query answering can be straightforwardly reduced to query entailment by considering the Boolean query obtained by substituting the distinguished variables of the query with the given tuple of individuals.

[6] When we view the query q as a relational structure in which the variables snd constants are the domain elements, then a match for q in an interpretation $\mathcal{I}$ is actually a *homomorphism* from q to $\mathcal{I}$ (cf. [22] for the case of CQs).

3 Query Answering by Rewriting in Lightweight DLs

We illustrate now the approach to query answering based on *query rewriting*, which was first introduced for answering UCQs over *DL-Lite* ontologies through the *PerfectRef* algorithm [13], and then extended to several other DLs [41,42,44,32], including also more expressive members of the *DL-Lite* family [3]. Such DLs share with *DL-Lite* some crucial properties that are necessary to make a rewriting based approach efficient. We illustrate now the approach for the case of UCQs over *DL-Lite*$_\mathcal{R}$ ontologies. We first recall that, in the case where the ontology is unsatisfiable, the answer to any UCQ is the set of all tuples of individuals appearing in the ontology. Therefore, we focus for now on the case where the ontology is satisfiable, and come back to satisfiability afterwards.

The key idea at the basis of the rewriting approach is to strictly separate the processing done with respect to the intensional level of the ontology (i.e., the TBox) from the processing done by taking into account the extensional level (i.e., the ABox, or data): (1) the query is processed and rewritten into a new query, based on the inclusion assertions in the TBox; (2) the TBox is discarded and the rewritten query is evaluated over the ABox, as if the ABox was a simple relational structure/database. More precisely, given a UCQ q over $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, the *positive inclusion assertions* of $\mathcal{T}$, i.e., those inclusion assertions that contain no negation in the right-hand side, are compiled into q, thus obtaining a new query q'. Such new query q' is then evaluated over $\mathcal{A}$, thus essentially reducing query answering to query evaluation over a database instance. Since the size of q' does not depend on the ABox, the data complexity of the whole query answering algorithm is the same as the data complexity of evaluating q'. A crucial property for *DL-Lite* is that, in the case where q is a UCQ, the query q' is also a UCQ. Hence, the data complexity of the whole query answering algorithm is in AC^0, which is the complexity of evaluating a FOL query over a relational database.

Canonical Model. The rewriting based approach relies in an essential way on the *canonical model property*, which holds for *DL-Lite* and for the horn variants of many other DLs [33,24]. Such property ensures that every satisfiable ontology $\mathcal{O}$ admits a canonical model that is the least constrained model among all models of $\mathcal{O}$, and that can be homomorphically embedded in all other models. This in turn implies that the canonical model correctly represents *all* the models of $\mathcal{O}$ with respect to the problem of answering positive queries (and in particular, UCQs). In other words, for every UCQ q, we have that $\mathsf{cert}(q, \mathcal{O})$ is contained in the result of the evaluation of q over the canonical model[7]. Intuitively, the canonical model for a *DL-Lite* ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains the ABox $\mathcal{A}$, and in addition might contain existentially implied objects, whose existence is enforced by the TBox assertions with $\exists P$ in the right-hand side. For example, if the TBox contains an assertion *Student* $\sqsubseteq \exists$*attends*, expressing that every student should attend something (presumably a course), and the ABox contains the fact *Student*(*john*), then the canonical model will contain a fact *attends*(*john*, o_n), where o_n is a newly introduced object.

[7] Note that, since the domain of the canonical model contains the individuals of the ABox, hence the evaluation of a query over such model can indeed return a set of individuals.

$$\begin{array}{rccc}
A_1 \sqsubseteq A_2 & \ldots, A_2(x), \ldots & \rightsquigarrow & \ldots, A_1(x), \ldots \\
\exists p \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, p(x, \_), \ldots \\
\exists p^- \sqsubseteq & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, p(\_, x), \ldots \\
A \sqsubseteq \exists p & \ldots, p(x, \_), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
A \sqsubseteq \exists p^- & \ldots, p(\_, x), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
\exists p_1 \sqsubseteq \exists p_2 & \ldots, p_2(x, \_), \ldots & \rightsquigarrow & \ldots, p_1(x, \_), \ldots \\
p_1 \sqsubseteq p_2 & \ldots, p_2(x, y), \ldots & \rightsquigarrow & \ldots, p_1(x, y), \ldots \\
\ldots & & &
\end{array}$$

Fig. 2. Rewriting of query atoms in *DL-Lite*$_{\mathcal{R}}$

First-Order Rewritability. We point out that the canonical model is in general infinite, hence it cannot be effectively computed in order to solve the query answering problem by actually evaluating the input query q over it. Instead, each CQ q_i in q is rewritten into a UCQ r_i in such a way that, whenever q_i has a match in some portion of the canonical model, then there will be a CQ among those in r_i that has a corresponding match in the ABox part of the canonical model. Informally, the rewriting algorithm initializes a set r of CQs with the CQs in the input query q, and processes each yet unprocessed query r_i in r by adding to r also all rewritings of r_i. For each atom α in r_i, it checks whether α can be rewritten by using one of the positive inclusions in the TBox, and if so, adds to r the CQ obtained from r_i by rewriting α. The rewriting of an atom uses a positive inclusion assertion as rewriting rule, applied from right to left, to compile away the knowledge represented by the positive inclusion itself. For example, using the inclusion $A_1 \sqsubseteq A_2$, an atom of the form $A_2(x)$ is rewritten to $A_1(x)$. Alternatively, we can consider this rewriting step as the application of standard resolution between the query and the inclusion $A_1 \sqsubseteq A_2$, viewed as the (implicitly universally quantified) formula $A_1(x) \rightarrow A_2(x)$. Other significant cases of rewritings of atoms are depicted in Figure 2, where each inclusion assertion in the left-most column accounts for rewriting the atom to the left of $\rightsquigarrow$ into the atom to the right of it. We have used "_" to denote a variable that occurs only once in the CQ (counting also occurrences in the head of the CQ). Besides rewriting atoms, a further processing step applied to r_i is to consider each pair of atoms α_1, α_2 occurring in the body of r_i that *unify*, and replace them with a single atom, also applying the most general unifier to the whole of r_i. In this way, variables that in r_i occur multiple times, might be replaced by an "_", and hence inclusion assertions might become applicable that were not so before the atom-unification step (cf. the rewriting rules in Figure 2 *requiring* the presence of "_").

The above presented rewriting technique realized through *PerfectRef*, allows us to establish that answering UCQs over satisfiable *DL-Lite*$_{\mathcal{R}}$ ontologies is *first-order rewritable*, i.e., the problem of computing certain answers over a satisfiable ontology can be reduced to the problem of evaluating a FOL query over the ABox of the ontology viewed as a database (with complete information). Specifically, let $\mathsf{rew}(q, \mathcal{T})$ denote the UCQ obtained as the result of applying *PerfectRef* to a UCQ q and *DL-Lite*$_{\mathcal{R}}$ TBox $\mathcal{T}$. Then, for every ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, we have that

$$\mathsf{cert}(q, \langle \mathcal{T}, \mathcal{A} \rangle) \;=\; \mathsf{ans}(\mathsf{rew}(q, \mathcal{T}), \mathcal{A})$$

where $\mathsf{ans}(\mathsf{rew}(q, \mathcal{T}), \mathcal{A})$ denotes the evaluation of the UCQ $\mathsf{rew}(q, \mathcal{T})$ over the ABox $\mathcal{A}$ viewed as a database (i.e., a first-order structure).

Ontology Satisfiability. The rewriting of a UCQ q with respect to a TBox $\mathcal{T}$ computed by *PerfectRef* depends only on the set of positive inclusion assertions in $\mathcal{T}$, while disjointness assertions (i.e., inclusion assertions containing a negated basic concept on the right-hand side) do not play any role in such a process. Indeed, the proof of correctness of *PerfectRef* [13], which is based on the canonical model property of *DL-Lite*$_\mathcal{R}$, shows that these kinds of assertions have to be considered only when verifying the ontology satisfiability. Once satisfiability is established, they can be ignored in the query rewriting phase. In fact, unsatisfiability of a *DL-Lite*$_\mathcal{R}$ ontology is due to the presence of disjointness assertions and their interaction with positive inclusions. Such interaction can itself be captured by constructing a Boolean UCQ encoding the violation of disjointness assertions, rewriting such a UCQ with respect to the positive inclusions, and checking whether its evaluation over the ABox returns true. This in turn shows that also the problem of checking satisfiability of a *DL-Lite*$_\mathcal{R}$ ontology is first-order rewritable [13].

Complexity of Query Evaluation. Summarizing the above results, and considering that evaluating a FOL query (and hence a UCQ) over a database is in AC^0 in data complexity, one obtains that answering UCQs over *DL-Lite*$_\mathcal{R}$ ontologies has the same data complexity as evaluating UCQs in plain databases. By analyzing the overall rewriting-based query answering technique, and by exploiting a correspondence between the *DL-Lite* family and FOL with unary predicates [3], we are able obtain also tight complexity bounds in the size of the TBox (*schema complexity*) and of the overall input (*combined complexity*).

Theorem 1 ([13,3]). *Answering UCQs over DL-Lite*$_\mathcal{R}$ *ontologies is in* AC^0 *in data complexity,* NLOGSPACE-*complete in schema complexity, and NP-complete in combined complexity.*

While the above results sound very encouraging from the theoretical point of view, there still remain significant challenges to be addressed to make rewriting based techniques effective also in real world scenarios, where the TBox and/or the data underlying the ABox are very large, and/or queries have a large number of atoms. Indeed, also in the case where one admits rewritings expressed in languages different from UCQs (e.g., arbitrary FOL queries, or non-recursive Datalog), it has recently been shown that the smallest rewritings can grow exponentially with the size of the query [28]. This has led to an intensive and sustained effort aimed at developing techniques to improve query answering over ontologies, such as alternative rewriting techniques [42,46], techniques combining rewriting with partial materialization of the extensional level [31], and various optimization techniques that take into account also extensional constraints on the underlying data, or a mapping layer to relational data sources, i.e., the so-called *Ontology-Based Data Access* (OBDA) setting [43,45]

4 Data Complexity for Query Entailment in Expressive DLs

When a DL contains (explicit or implicit) forms of disjunction, there is no single model representing all possible models for the purpose of query answering. Hence, approaches that exploit the canonical model property, e.g., those based on query rewriting, are not directly applicable in this case. We illustrate now an alternative query answering technique that builds upon the *tableaux-based techniques* that have proved very successful for reasoning in expressive DLs [6,30,29,37].

Tableaux Algorithms for Ontology Reasoning. We illustrate first the idea underlying the use of tableaux algorithms for checking satisfiability of a DL ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, and show then how this approach can be adapted for query entailment. The tableaux algorithm tries to build a model of $\mathcal{O}$ by starting from the assertions in $\mathcal{A}$, and completing them according to what is required by $\mathcal{T}$. In doing so it builds non-deterministically a forest-shaped relational structure (hence, the algorithm actually maintains a set of structures), that we call here *completion forest*. The structure is forest-shaped, since each individual in the ABox $\mathcal{A}$ is the root of a tree generated by applying tableaux-style *expansion rules* to the facts in the completion forest. Essentially, each rule is associated to one of the constructs of the DL, has a precondition, expressed as one or more facts to which the rule is applied, and possibly comes with additional conditions related to applicability of the rule, or to blocking (necessary to ensure termination). As the result of the rule application, the set of facts in the completion forest is expanded, or more in general, changed, possibly by introducing new individuals. To deal, e.g., with incompleteness caused by the presence of disjunction, or with at-most restrictions that might require the identification of individuals, some of the rules are non-deterministic, and cause the generation of more than one new completion forest from the current one.

When no more rules can be applied, and no obvious contradiction (called a *clash*) is present in the current completion forest, the algorithm terminates, and the clash-free completion forest witnesses a model of the ontology $\mathcal{O}$ (soundness of the algorithm). Instead, when each of the non-deterministically generated completion forests contains a clash, it means that $\mathcal{O}$ does not admit any model, hence is unsatisfiable (completeness of the algorithm). In order to avoid infinite repetition of rule application, and hence ensure termination of the algorithm, suitable *blocking conditions* need to be applied. Intuitively, the nodes of a completion forest are labeled with sets of concepts, and a rule application is considered blocked for a node x if in the tree there is a predecessor of x labeled in a way that is "compatible" with x. The precise notion of "compatibility" between two nodes depends on the constructs of the considered DL, and might also involve looking at pairs of adjacent nodes, rather than at single nodes, see, e.g., [6].

Tableaux Algorithms for Query Entailment. As shown for the first time in [35] for the system CARIN, tableaux algorithms can be adapted to deal also with query entailment. We illustrate here the approach presented in [39] for entailment of a PQ q in an ontology $\mathcal{O}$ expressed in one of $\mathcal{ALCOIH}$, $\mathcal{ALCOQH}$, or $\mathcal{ALCIQH}$. As for the case of satisfiability, the technique makes use of completion forests, each of which is meant to capture a set of models of the ontology $\mathcal{O}$. Actually, at each step of the algorithm, each of the models of $\mathcal{O}$ is represented by one of the completion forests in the set maintained non-deterministically by the algorithm. More specifically, when a tableaux rule is applied to a completion forest F, each of the models represented by F is preserved in one of the completion forests generated as a result of the rule application. Therefore, checking whether $O \models q$ equals checking whether $F \models q$, for each completion forest F. The crux of the correctness of the technique lies in the fact that, for largely enough expanded F, one can check whether $F \models q$ effectively via a syntactic mapping of the variables in q to the nodes in F. Thus, to witness that $\mathcal{O} \not\models q$, it is sufficient to (non-deterministically) construct a large enough forest F to which q cannot be mapped.

As customary with tableaux-style algorithms, the algorithm makes use of suitable blocking conditions on the rules to ensure termination of forest expansion. The blocking conditions adopted for $\mathcal{ALCOIH}$, $\mathcal{ALCOQH}$, and $\mathcal{ALCIQH}$ are inspired by those in [35] for CARIN, but are able handle on the one hand nominals present in $\mathcal{ALCOIH}$ and $\mathcal{ALCOQH}$, and on the other hand the fact that $\mathcal{ALCIQH}$ does not have the *finite model property*. Lack of this property means that reasoning with respect to arbitrary models is different from reasoning with respect to finite models only, and implies that $\mathcal{O} \not\models q$ might hold, but this might be witnessed only by an infinite model. As a consequence, expansion forests cannot be considered themselves as models of $\mathcal{O}$, but rather are finite representations of possibly infinite structures obtained by unraveling the completion forest. A further complication comes from the fact that in the blocking conditions of the tableaux rules it is not sufficient anymore to consider single nodes (or pairs of adjacent nodes) as for satisfiability [6]. Instead, one needs to search in the completion forest for the repetition of subtrees, whose depth depends on the number of atoms in the query q. This leads to an additional exponential blowup in the computational complexity of the algorithm with respect to the tableaux algorithms for satisfiability in the same logics.

We also note that [39] presents a single algorithm for checking query entailment in $\mathcal{ALCOIH}$, $\mathcal{ALCOQH}$, and $\mathcal{ALCIQH}$, which includes tableaux rules for all of the constructs in the three logics. However, due to the subtle interaction between nominals, inverse roles, and number restrictions, termination of the algorithm is guaranteed only for TBoxes expressed in $\mathcal{ALCOIH}$, $\mathcal{ALCOQH}$, or $\mathcal{ALCIQH}$.

Complexity of Query Entailment. Interestingly, while the above described tableaux algorithm for checking $\mathcal{O} \models q$ is not computationally optimal in combined complexity, a careful analysis shows that the construction of completion forests, and the check whether q can be mapped to each such forest, can both be carried out by a non-deterministic algorithm that runs in polynomial time in the size of the ABox (and the number of individuals appearing in nominals). Hence, the overall algorithm is coNP in data complexity. This is also computationally optimal, since checking query entailment for CQs over an ontology whose TBox contains a single assertion of the form $A_1 \sqsubseteq A_2 \sqcup A_3$ is already coNP-hard in data complexity [14].

Theorem 2 ([39]). *Given an $\mathcal{ALCOIH}$, $\mathcal{ALCOQH}$, or $\mathcal{ALCIQH}$ ontology $\mathcal{O}$ and a PQ q, deciding whether $O \models q$ is:*

- *in co*N3EXPTIME *in combined complexity.*
- *in co*N2EXPTIME *in combined complexity for a fixed* q *(under the assumption that numbers appearing in number restrictions are encoded in unary).*
- *coNP-complete in data complexity.*

5 Query Entailment in Very Expressive DLs

We address now query entailment for the case where the ontology and/or the query may contain roles built as regular expressions over direct and inverse roles, or their Boolean combinations.

Automata Techniques for Reasoning over Ontologies. For many very expressive DLs, including those which allow for the use of regular expressions over roles, the standard reasoning task of checking concept satisfiability (possibly with respect to a TBox) is naturally solvable by tree-automata, thanks to the *tree model property* of such logics: each satisfiable concept C has a tree-shaped model [50,52] in which nodes are labeled with sets of concepts, and adjacent nodes in the tree are connected by one or more roles. Intuitively, such a tree-shaped model is obtained by unraveling an arbitrary model, introducing new nodes in the tree whenever the same node is encountered multiple times during the unraveling. Hence, one can construct a tree-automaton that accepts a tree representing a tree-shaped model, by naturally encoding in the transition function of the automaton the conditions that the DL constructs impose on adjacent nodes of the model/tree. Checking for the existence of a model amounts to checking for non-emptiness of the tree-automaton. A crucial observation is that, even for those logics that have the finite model property, the unraveling process produces an infinite tree, so that we need to resort to automata on *infinite* trees [48].

When also an ABox $\mathcal{A}$ is present this approach fails, since the assertions in $\mathcal{A}$ may arbitrarily connect individuals, and thus destroy the tree-structure. On the other hand, while a satisfiable $\mathcal{DL}$ ontology $\mathcal{O} = \langle \mathcal{A}, \mathcal{T} \rangle$ may lack a tree-shaped model, it always has a forest-shaped *canonical model*, in which the individuals in $\mathcal{A}$ can be arbitrarily connected, but each individual is the root of a tree-shaped model of $\mathcal{T}$. This property is usually sufficient to adapt algorithms for concept satisfiability so as to decide also ontology satisfiability. In particular, automata-based algorithms have been adapted, e.g., using the *pre-completion* technique [49], in which after a reasoning step on the ABox, automata are used to verify the existence of a tree-shaped model rooted at each ABox individual.

Automata Techniques for Query Entailment. However, a pre-completion based approach would not lend itself well for query entailment, where one needs to account also for the interaction between the variables in query atoms and the ABox individuals to which these variables have to be mapped. This holds especially in the case where the query itself might contain atoms that are regular expressions over roles, as in RPQs and their extensions. Therefore, we discuss here briefly a different approach to query entailment for very expressive DLs and queries, based on the idea of representing forest-shaped interpretations directly as trees in which the root is a dummy node, and all individuals appearing in the ABox and in nominals form the children of the root. From each of these first-level nodes, a possibly infinite tree departs. Adopting this kind of representation allows us to deal in a uniform way using tree automata both with the TBox and ABox constituting the ontology, and with the query [19,20,21].

Specifically, we illustrate an approach to check query entailment $\mathcal{O} \models q$, that is applicable when q is a P2RPQs and $\mathcal{O}$ is an ontology expressed in any sublanguage of

$\mathcal{DL}$ in which only two of the three constructs of nominals ($\mathcal{O}$), inverse roles ($\mathcal{I}$), and number restrictions ($\mathcal{Q}$) are present [20]. We use $\mathcal{DL}^-$ to denote such a logic. To decide whether $\mathcal{O} \models q$, it is sufficient to decide whether $\mathcal{O}$ has a tree-shaped (canonical) model in which q has no match. To check this using tree automata, we build an automaton $\mathbf{A}_{\mathcal{O}\not\models q}$ that accepts all trees that represent a model of $\mathcal{O}$ in which q has no match. Roughly speaking, $\mathbf{A}_{\mathcal{O}\not\models q}$ is obtained by intersecting two automata:

– $\mathbf{A}_{\mathcal{O}}$, which is a tree automaton that accepts the trees representing a model of $\mathcal{O}$. We make use of two-way alternating tree automata [51]: on the one hand, such automata can traverse a tree both downwards and upwards, which turns out to be convenient to deal with inverse roles; on the other hand, such automata are alternating, which means that they are equipped both with $\wedge$-transitions and with $\vee$-transitions, which allows one to naturally encode in the transition function of the automaton the structural conditions imposed by concept and role expressions.
– $\mathbf{A}_{\neg q}$, which accepts the trees representing an interpretation that admits no match for q. To obtain $\mathbf{A}_{\neg q}$, we first construct an automaton $\mathbf{A}_q$ that accepts a tree T if and only if q has a match in the interpretation represented by T. To construct $\mathbf{A}_q$, we need to treat the existential variables appearing in q as additional concept symbols, and represent them explicitly in the tree accepted by the automaton. Then, to construct $\mathbf{A}_{\neg q}$, we need to project away such additional symbols, before complementing the automaton, which results in an (inevitable) exponential blowup.

Hence, deciding query entailment reduces to checking whether $\mathbf{A}_{\mathcal{O}\not\models q}$ accepts the empty language.

The details of the constructions of the above described automata are quite involved. For the constructions, the proof of their correctness, and the computational complexity analysis, we refer to [21] for the case when $\mathcal{DL}^-$ does not include nominals (which corresponds to the DL $\mathcal{ZIQ}$ of [20]), and to [20] for the cases when $\mathcal{DL}^-$ includes nominals ($\mathcal{ZOI}$ and $\mathcal{ZOQ}$).

Theorem 3 ([19,21,20]). *Given an $\mathcal{DL}^-$ ontology $\mathcal{O}$ and a PQ q, deciding whether $O \models q$ is in* 2EXPTIME *in combined complexity (under the assumption that numbers appearing in number restrictions are encoded in unary).*

The above bound is tight, since query entailment is already 2EXPTIME-hard for the following cases:

– CQs over ontologies expressed in $\mathcal{ALCI}$ [36] or $\mathcal{ALCH}_{reg}$ [23];
– CRPQs or PQs over ontologies expressed in $\mathcal{ALC}$ [8].

Notice that the automata-theoretic approach above does not provide us any bound on data complexity (that is better than the one for combined complexity). In fact, it remains to be investigated whether in this approach it is possible to single out the contribution coming from the ABox and the nominals in the construction of the automata and the final emptiness check.

The above automata-based technique does also not work for the case of full $\mathcal{DL}$, i.e., when the logic contains both nominals, inverses, and number restrictions, since in such a case the tree model property fails, and tree automata do not seem suitable anymore as a technical tool. In fact, decidability of entailment for PQs over $\mathcal{ALCOIQ}$ ontologies has been established in [27] using model theoretic arguments, which do not provide any complexity upper bound. However, the problem is still open for logics including also regular expressions over roles (or alternatively, transitive roles also in the query, in line with what can be expressed in the Web Ontology Language OWL [7]).

Finally, we mention recent work that has considered the problem of query entailment also over lightweight DLs, for variants of queries containing regular expressions [9], and for their extension with nesting [8].

6 Conclusions

In this work, we have provided an overview of three prominent techniques that have been used in recent years to address the challenging problem of query answering and query entailment in DLs. Specifically, we have discussed: *(1)* a technique based on query rewriting suitable for UCQs over lightweight DLs, which provides optimal complexity bounds due to first-order rewritability; *(2)* a technique based on tableaux suitable for PQs over expressive DLs, which provides optimal bounds in data complexity, but not in combined complexity, and is not able to deal with regular expressions over roles in the ontology or the query; *(3)* a technique based on automata on infinite trees, which is able to deal with very expressive ontology and query languages containing regular expressions over roles, and provides optimal complexity bounds in combined complexity, but not in data complexity.

The research on query answering and query entailment is still very active, and the problem continues to provide challenges. On the one hand, from the theoretical point of view several decidability and complexity questions are still open. On the other hand, more work is required to implement query answering and query entailment algorithms for the more expressive ontology and query languages. And both for lightweight and for expressive languages, improvements in efficiency are needed, so at to make these techniques usable in real world scenarios.

Acknowledgments. This work has been partially supported by the EU IP project Optique (*Scalable End-user Access to Big Data*), grant agreement n. FP7-318338.

References

1. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann (2000)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co. (1995)
3. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)

5. Baader, F., Sattler, U.: Expressive number restrictions in description logics. J. of Logic and Computation 9(3), 319–350 (1999)
6. Baader, F., Sattler, U.: Tableau algorithms for description logics. In: Dyckhoff, R. (ed.) TABLEAUX 2000. LNCS (LNAI), vol. 1847, pp. 1–18. Springer, Heidelberg (2000)
7. Bao, J., et al.: OWL 2 Web Ontology Language document overview, W3C Recommendation, World Wide Web Consortium, 2nd edn. (December 2012), `http://www.w3.org/TR/owl2-overview/`
8. Bienvenu, M., Calvanese, D., Ortiz, M., Simkus, M.: Nested regular path queries in description logics. In: Proc. of the 14th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2014). AAAI Press (2014)
9. Bienvenu, M., Ortiz, M., Simkus, M.: Conjunctive regular path queries in lightweight description logics. In: Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence, IJCAI 2013 (2013)
10. Buneman, P., Davidson, S., Hillebrand, G., Suciu, D.: A query language and optimization technique for unstructured data. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 505–516 (1996)
11. Calvanese, D., De Giacomo, G.: Expressive description logics. In: Baader, et al. (eds.) [4], ch. 5, pp. 178–218
12. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and databases: The DL-Lite approach. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009)
13. 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)
14. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. Artificial Intelligence 195, 335–360 (2013)
15. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 1998), pp. 149–158 (1998)
16. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Containment of conjunctive regular path queries with inverse. In: Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000), pp. 176–185 (2000)
17. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Rewriting of regular expressions and regular path queries. J. of Computer and System Sciences 64(3), 443–465 (2002)
18. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Reasoning on regular path queries. SIGMOD Record 32(4), 83–92 (2003)
19. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proc. of the 22nd AAAI Conf. on Artificial Intelligence (AAAI 2007), pp. 391–396 (2007)
20. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009), pp. 714–720 (2009)
21. Calvanese, D., Ortiz, M., Eiter, T.: Answering regular path queries in expressive description logics via alternating tree-automata. Information and Computation 237, 12–55 (2014)
22. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Proc. of the 9th ACM Symp. on Theory of Computing (STOC 1977), pp. 77–90 (1977)
23. Eiter, T., Lutz, C., Ortiz, M., Šimkus, M.: Query answering in description logics with transitive roles. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009), pp. 759–764 (2009)

24. Eiter, T., Ortiz, M., Simkus, M., Tran, T.K., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012). AAAI Press (2012)
25. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theoretical Computer Science 336(1), 89–124 (2005)
26. Florescu, D., Levy, A., Suciu, D.: Query containment for conjunctive queries with regular expressions. In: Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 1998), pp. 139–148 (1998)
27. Glimm, B., Rudolph, S.: Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend. J. of Artificial Intelligence Research 39, 429–481 (2010)
28. Gottlob, G., Kikot, S., Kontchakov, R., Podolskii, V.V., Schwentick, T., Zakharyaschev, M.: The price of query rewriting in ontology-based data access. Artificial Intelligence 213, 42–59 (2014)
29. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67 (2006)
30. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. J. of Automated Reasoning 39(3), 249–276 (2007)
31. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010), pp. 247–257 (2010)
32. Krisnadhi, A., Lutz, C.: Data complexity in the $\mathcal{EL}$ family of description logics. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 333–347. Springer, Heidelberg (2007)
33. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for horn description logics. In: Proc. of the 22nd AAAI Conf. on Artificial Intelligence (AAAI 2007), pp. 452–457 (2007)
34. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002), pp. 233–246 (2002)
35. Levy, A.Y., Rousset, M.C.: Combining Horn rules and description logics in CARIN. Artificial Intelligence 104(1-2), 165–209 (1998)
36. Lutz, C.: Inverse roles make conjunctive queries hard. In: Proc. of the 20th Int. Workshop on Description Logic (DL 2007). CEUR Electronic Workshop Proceedings, vol. 250, pp. 100–111 (2007), `http://ceur-ws.org/`
37. Lutz, C., Milicic, M.: A tableau algorithm for description logics with concrete domains and general tboxes. J. of Automated Reasoning Special Issue on Automated Reasoning with Analytic Tableaux and Related Methods 38(1-3), 227–259 (2007)
38. Ortiz, M.: Ontology based query answering: The story so far. In: Proc. of the 7th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW 2013). CEUR Electronic Workshop Proceedings, vol. 1087 (2013), `http://ceur-ws.org/`
39. 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)
40. Ortiz, M., Šimkus, M.: Reasoning and query answering in description logics. In: Eiter, T., Krennwallner, T. (eds.) Reasoning Web 2012. LNCS, vol. 7487, pp. 1–53. Springer, Heidelberg (2012)
41. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-lite. In: Proc. of the 22nd Int. Workshop on Description Logic (DL 2009). CEUR Electronic Workshop Proceedings, vol. 477 (2009), `http://ceur-ws.org/`
42. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. of Applied Logic 8(2), 186–209 (2010)

43. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012), pp. 308–318 (2012)
44. Rosati, R.: On conjunctive query answering in $\mathcal{EL}$. In: Proc. of the 20th Int. Workshop on Description Logic (DL 2007). CEUR Electronic Workshop Proceedings, vol. 250, pp. 451–458 (2007), `http://ceur-ws.org/`
45. Rosati, R.: Prexto: Query rewriting under extensional constraints in DL-Lite. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 360–374. Springer, Heidelberg (2012)
46. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010), pp. 290–300 (2010)
47. Schmolze, J.G.: Terminological knowledge representation systems supporting n-ary terms. In: Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 1989), pp. 432–443 (1989)
48. Thomas, W.: Automata on infinite objects. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, ch. 4, pp. 133–192. Elsevier Science Publishers (1990)
49. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Ph.D. thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany (2001)
50. Vardi, M.Y.: Why is modal logic so robustly decidable. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 31, pp. 149–184. American Mathematical Society (1997)
51. Vardi, M.Y.: Reasoning about the past with two-way automata. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 628–641. Springer, Heidelberg (1998)
52. Vardi, M.Y., Wolper, P.: Automata-theoretic techniques for modal logics of programs. J. of Computer and System Sciences 32, 183–221 (1986)