Answering Expressive Path Queries over Lightweight DL Knowledge Bases *

Meghyn Bienvenu[1], Magdalena Ortiz[2], and Mantas Šimkus[2]

[1] LRI - CNRS & Université Paris Sud
[2] Institute of Information Systems, Vienna University of Technology

Abstract. We establish tight complexity bounds for answering an extension of conjunctive 2-way regular path queries over $\mathcal{EL}$ and DL-Lite knowledge bases.

1 Introduction

It has been extensively argued in the description logic (DL) community that answering queries over ABoxes in the presence of ontological constraints formulated in a DL TBox is a fundamental reasoning service. In databases, similar attention has been paid to the related problem of querying graph databases, which are relational databases where only unary and binary predicates occur, or in other words, node- and edge-labeled graphs [8, 3]. The relevance of both problems lies in the fact that in many application areas data can be naturally modeled as an ABox or a graph database. This applies, in particular, to XML data on the web, including RDF datasets. While the two communities share some common research goals, the research agendas they have pursued differ significantly. In the DL community, the focus has been on designing efficient algorithms for answering (plain) conjunctive queries in the presence of expressive ontological constraints. By contrast, work on graph databases typically does not consider ontological knowledge, but instead aims at supporting expressive query languages, like regular path queries (RPQs) and their extensions, which enable sophisticated navigation of paths.

This paper aims to help bridge this gap, by considering an expressive extension of RPQs, and providing algorithms and precise complexity bounds for the $\mathcal{EL}$ and DL-Lite families of lightweight DLs. We build on *conjunctive (2-way) regular path queries* (C2RPQs), which simultaneously extend plain conjunctive queries (CQs) and basic RPQs: they allow conjunctions of atoms that can share variables in arbitrary ways, where the atoms may contain regular expressions that navigate the arcs of the database (roles) in both directions. C2RPQs are one of the most expressive and popular languages for querying graph databases. These queries have already been studied for some DLs. In particular, automata-based algorithms have been proposed for the very expressive DLs $\mathcal{ZIQ}$, $\mathcal{ZIO}$, and $\mathcal{ZOQ}$ [5, 6], for which query answering is 2-ExpTime hard. Even in data complexity, that is, when the query and ontology are assumed fixed, these algorithms need exponential time. More recently, algorithms for answering C2RPQs were proposed in [11] for Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$. They are polynomial in data complexity but still ExpTime in combined, which is worst-case optimal for these

* This work partially supported by the Austrian Science Fund (FWF) grants P20840 and T515.

logics. For prominent lightweight DLs like the DL-Lite [4] and $\mathcal{EL}$ [2], which underly the OWL 2 profiles, queries with regular paths had not been explored and many questions remained open, like whether algorithms that require only polynomial space are possible. In DL-Lite, FO-rewritability and AC_0 data complexity are clearly lost, since we can express reachability, but it was not known whether P-hardness was avoidable. In this paper, we answer these questions by providing precise complexity bounds.

We propose an extension of C2RPQs that we call *conjunctive (2-way) regular path queries with complex labels*, abbreviated ℓ-C2RPQs. To illustrate its expressiveness, we consider a graph representation of the *Mathematics Genealogy Project (MGP)* database, which contains about 160K historic records of advisor relationships of PhD holders in mathematics and related disciplines. We use nodes for mathematicians, theses, topics of research, and universities. Nodes and edges are labeled with concepts (unary relations) and roles (binary relations), respectively. Figure 1 depicts a fragment of such a graph.

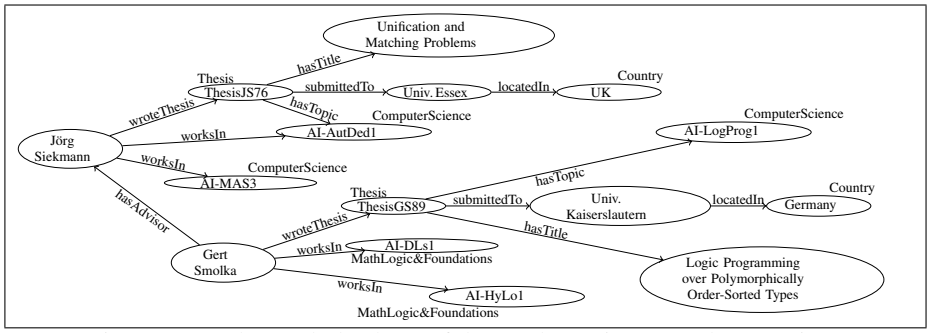

Fig. 1: Example graph database of the Mathematics Genealogy Project

An ontology containing the axioms in Fig. 2 can be used to express, for example, that a person that works in computer science or that wrote a doctoral thesis in computer science is a computer scientist (1,2). In a similar way we can define other specialities such as biologists (3,4), logicians (5,6), physicists, etc. We group the first level subjects of the Mathematics Subject Classification (MSC) used in the MGP database into their 5 major areas (7–11), and these major areas into subjects (12–16).

(1)	$\exists worksOn.CompSci \sqsubseteq CompScientist$		
(2)	$\exists wroteThesis.(\exists hasTopic.CompSci) \sqsubseteq CompScientist$		
(3)	$\exists worksOn.Biology\&NaturalSciences \sqsubseteq Biologist$		
(4)	$\exists wroteThesis.(\exists hasTopic.Biology\&NaturalSciences) \sqsubseteq Biologist$		
(5)	$\exists worksOn.MathLogic\&Foundns \sqsubseteq Logician$		
(6)	$\exists wroteThesis.(\exists hasTopic.MathLogic\&Foundns) \sqsubseteq Logician$		
(7)	$MathLogic\&Foundns \sqsubseteq General\&Foundns$	(12)	$General\&Foundns \sqsubseteq Subject$
(8)	$Geometry \sqsubseteq Geometry\&Topology$	(13)	$DiscreteMath\&Algebra \sqsubseteq Subject$
(9)	$CompSci \sqsubseteq AppliedMath\&Other$	(14)	$Analysis \sqsubseteq Subject$
(10)	$Biology\&NaturalSciences \sqsubseteq AppliedMath\&Other$	(15)	$Geometry\&Topology \sqsubseteq Subject$
(11)	$Physics \sqsubseteq AppliedMath\&Other$	(16)	$AppliedMath\&Other \sqsubseteq Subject$

Fig. 2: An example MGP ontology

In RPQs, C2RPQs and similar languages, regular paths usually talk about the arc labels only, and do not allow to verify conditions on the node labels. For example, one can use the expression $hasAdvisor^* \circ WroteThesis \circ hasTopic$ to navigate arbitrarily long chains of advisors and visit their thesis topics, but we cannot look at the subjects and thesis topics and impose conditions on them. This limitation is not so significant for

graph databases, as arc labels play a more prominent role and are often used to simulate node labels. In the DL setting, by contrast, concepts are crucial and should be treated as first-class citizens. For this reason, we add to C2RPQs the ability to talk about combinations of concept and roles that appear along a path. In our language, we can use the expression $(hasAdvisor, Logician \vee CompScientist)^* \circ WroteThesis \circ (hasTopic, Geometry)$ to navigate a chain of advisors that are computer scientist or logicians, until we reach one that wrote a doctoral thesis in Geometry. Note that this query could be expressed (less succinctly) using the *test* operator (cf. [6]): $(hasAdvisor \circ Logician? \cup hasAdvisor \circ CompScientist?)^* \circ WroteThesis \circ hasTopic \circ Geometry?$. However, our language is slightly more expressive (for DLs without role conjunction), since we can navigate a chain of people that are both advisors and coauthors using $(hasAdvisor \wedge coAuthor)^*$.

In this paper, we show that answering these queries over $\mathcal{EL}$ and DL-Lite knowledge bases is PSPACE-complete, but drops to NP if we consider DL-Lite$_{\mathsf{RDFS}}$. For data complexity, the problem is NLSPACE-complete for DL-Lite and P-complete for $\mathcal{EL}$.

2 Preliminaries

We briefly recall the syntax of DL-Lite$_{\mathcal{R}}$ [4] and $\mathcal{ELH}$ [2] (and relevant sublogics). As usual, we assume sets $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ of concept names, role names, and individuals. We will use $\overline{\mathsf{N_R}}$ to refer to $\mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$, and if $R \in \overline{\mathsf{N_R}}$, we use R^- to mean r^- if $R = r$ and r if $R = r^-$. An ABox is a set of assertions of the form $A(b)$ or $r(b, c)$, where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $b, c \in \mathsf{N_I}$. A TBox is a set of inclusions, whose form depends on the DL in question. In DL-Lite, inclusions take the form $B_1 \sqsubseteq (\neg)B_2$, where each B_i is either A (where $A \in \mathsf{N_C}$) or $\exists R$ (where $R \in \overline{\mathsf{N_R}}$). DL-Lite$_{\mathcal{R}}$ additionally allows role inclusions of the form $R_1 \sqsubseteq (\neg)R_2$, where $R_1, R_2 \in \overline{\mathsf{N_R}}$. DL-Lite$_{\mathsf{RDFS}}$ is obtained from DL-Lite$_{\mathcal{R}}$ by disallowing inclusions which contain negation or have existential concepts ($\exists R$) on the right-hand side. In $\mathcal{EL}$, inclusions have the form $C_1 \sqsubseteq C_2$, where C_1, C_2 are complex concepts constructed as follows: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$. The DL $\mathcal{ELH}$ additionally allows role inclusions of the form $r \sqsubseteq s$, where $r, s \in \mathsf{N_R}$. A knowledge base (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

As usual, the semantics is based upon interpretations, which take the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ maps each $a \in \mathsf{N_I}$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each $A \in \mathsf{N_C}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each $r \in \mathsf{N_R}$ to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is straightforwardly extended to general concepts and roles, e.g. $(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$ and $(\exists r.C)^{\mathcal{I}} = \{c \mid \exists d : (c, d) \in r^{\mathcal{I}}, d \in C^{\mathcal{I}}\}$. $\mathcal{I}$ satisfies $G \sqsubseteq H$ if $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$; it satisfies $A(a)$ (resp. $r(a, b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$). $\mathcal{I}$ is a *model* of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I}$ satisfies all inclusions in $\mathcal{T}$ and assertions in $\mathcal{A}$.

To simplify the presentation, we will assume that $\mathcal{ELH}$ TBoxes are *normalized*, meaning that all concept inclusions are of one of the following forms:

$$\top \sqsubseteq A \qquad A \sqsubseteq B \qquad A \sqsubseteq \exists r.B \qquad B_1 \sqcap B_2 \sqsubseteq A \qquad \exists r.B \sqsubseteq A$$

with A, B, B_1, B_2 concept names. It is well-known that for every $\mathcal{ELH}$ TBox $\mathcal{T}$, one can construct in polynomial time a normalized $\mathcal{ELH}$ TBox $\mathcal{T}'$ that uses fresh concept names such that $\mathcal{T}' \models \mathcal{T}$ and every model of $\mathcal{T}$ can be expanded to a model of $\mathcal{T}'$.

For convenience, we introduce a set of *basic concepts*, denoted BC, defined as follows: $\mathsf{BC} = \mathsf{N_C} \cup \{\exists R \mid R \in \overline{\mathsf{N_R}}\}$ for DL-Lite$_{\mathcal{R}}$, and $\mathsf{BC} = \mathsf{N_C}$ for $\mathcal{ELH}$.

Canonical Models We recall the definition of canonical models for DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ KBs. For both logics, the domain of the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ for a KB $(\mathcal{T},\mathcal{A})$ will consist of *paths* of the form $aR_1C_1 \dots R_nC_n$ ($n \geq 0$), where $a \in \mathsf{Ind}(\mathcal{A})$, each C_i is a basic concept, and each R_i a (possibly inverse) role. When $\mathcal{T}$ is a DL-Lite$_\mathcal{R}$ TBox, the domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ contains exactly those paths $aR_1\exists R_1^- \dots R_n\exists R_n^-$ which satisfy:

- if $n \geq 1$, then $\mathcal{T},\mathcal{A} \models \exists R_1(a)$;
- for $1 \leq i < n$, $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_i^- \neq R_{i+1}$.

When $\mathcal{T}$ is a (normalized) $\mathcal{ELH}$ TBox, the domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ contains exactly those paths $ar_1A_1 \dots r_nA_n$ for which each $r_i \in \mathsf{N_R}$, and:

- if $n \geq 1$, then $\mathcal{T},\mathcal{A} \models \exists r_1.A_1(a)$;
- for $1 \leq i < n$, $\mathcal{T} \models A_i \sqsubseteq \exists r_{i+1}.A_{i+1}$.

We denote the last concept in a path p by $\mathsf{tail}(p)$, and define $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ by taking:

$$
\begin{aligned}
a^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= a \text{ for all } a \in \mathsf{Ind}(\mathcal{A}) \\
A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T},\mathcal{A} \models A(a)\} \cup \{p \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T} \models \mathsf{tail}(p) \sqsubseteq A\} \\
r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{(a,b) \mid r(a,b) \in \mathcal{A}\} \cup \\
&\quad \{(p_1,p_2) \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \times \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \mid p_2 = p_1 \cdot S\,C \text{ and } \mathcal{T} \models S \sqsubseteq r\} \cup \\
&\quad \{(p_2,p_1) \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \times \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \mid\ p_2 = p_1 \cdot S\,C \text{ and } \mathcal{T} \models S \sqsubseteq r^-\}
\end{aligned}
$$

Note that $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is composed of a core consisting of the ABox individuals and an *anonymous part* consisting of (possibly infinite) trees rooted at ABox individuals. We will use $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_e$ to denote the submodel of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ obtained by restricting the universe to paths having e as a prefix.

Regular Languages We assume the reader is familiar with regular languages, represented either by regular expressions or nondeterministic finite state automata (NFAs). An NFA over an *alphabet* Σ is a tuple $\alpha = \langle S, \Sigma, \delta, s_0, F\rangle$, where S is a finite set of *states*, $\delta \subseteq S \times \Sigma \times S$ the *transition relation*, $s_0 \in S$ the *initial state*, and $F \subset S$ the set of *final states*. We use $L(\alpha)$ to denote the language defined by an NFA α, and when the way a regular language is represented is not relevant, we denote it simply by L.

3 Conjunctive Regular Path Queries with Complex Labels

We now formally introduce our query language.

Definition 1. *By $\mathcal{B}(S)$ we denote the set of all (positive) Boolean formulas built from the symbols in $S \cup \{\mathbf{true}, \mathbf{false}\}$ using the connectives $\wedge$ and $\vee$. A conjunctive (two-way) regular path query with complex labels (abbreviated to ℓ-C2RPQ) has the form $q(\boldsymbol{x}) = \exists \boldsymbol{y}\ \varphi$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are tuples of variables, and φ is a conjunction of atoms of the following forms:*

(i) $\beta(t)$, where $\beta \in \mathcal{B}(\mathsf{N_C})$ and $t \in \mathsf{N_I} \cup \boldsymbol{x} \cup \boldsymbol{y}$, and
(ii) $L(t,t')$, where L is (an NFA or regular expression defining) a regular language over $\mathcal{B}(\overline{\mathsf{N_R}}) \times \mathcal{B}(\mathsf{N_C})$, and $t,t' \in \mathsf{N_I} \cup \boldsymbol{x} \cup \boldsymbol{y}$.

As usual, variables and individuals are called terms*, and the variables in $\boldsymbol{x}$ are called* answer variables*. A query with no answer variables is called* Boolean*.*

If all atoms of type (i) are of the form $A(t)$ with $A \in \mathsf{N_C}$, and the regular languages L in atoms of type (ii) comprise only symbols of the form $(R, \mathbf{true})$ with $R \in \overline{\mathsf{N_R}}$, then

q is called a conjunctive (two-way) regular path query *(C2RPQ). Conjunctive one-way regular path queries with complex labels (ℓ-CRPQs) and conjunctive one-way regular path queries (CRPQs) are defined analogously, the only difference being that they use formulas from $\mathcal{B}(\mathsf{N_R})$ instead of $\mathcal{B}(\overline{\mathsf{N_R}})$ in atoms of type (ii). Conjunctive queries (CQs) are C2RPQs where all atoms of type (ii) have the form $(R, \mathbf{true})(t, t')$ with $R \in \overline{\mathsf{N_R}}$.*

Example 1. For readability, we write symbols $(R, \mathbf{true}) \in \mathcal{B}(\overline{\mathsf{N_R}}) \times \mathcal{B}(\mathsf{N_C})$ simply as R. Recall the MGP database. The query $q_1(x, y)$ in Fig. 3 searches for pairs of computer scientists that have a biologist as common academic ancestor, and such that there is a physicist on that path. It returns, among others, Jack Minker and Edmund Clarke, who have the biologist and mathematician Johann Bernoulli (1667–1748) as common ancestor on a path including the physicist and mathematician Joseph Fourier (1768 – 1830); as well as Gert Smolka and Georg Gottlob, who have as ancestors Nikolaus Poda von Neuhaus, an Austrian entomologist (1723 - 1798), and the physicist Ludwig Boltzmann (1844 – 1906). The query $q_2(x, y)$ searches for a common academic ancestor x of Robert Kowalski and Franz Baader, together with the country y where the ancestor's thesis was defended; it requires all ancestors on the path to be computer scientists and logicians. It returns one tuple: (Bernard Meltzer, UK). The query $q_3(x)$ is similar but we require x to be a biologist or physicist, and additionally allow physicists along the path. This query retrieves 8 people, going back to Gabriel Gruber (1740–1805), a Jesuit priest, philosopher, mathematician and professor of physics.

$$
\begin{aligned}
q_1(x,y) = {} & CompScientist \vee Logician(x), CompScientist \vee Logician(y), \\
& \big[hasAdvisor^* \circ (hasAdvisor, Physicist) \circ hasAdvisor^* \circ (hasAdvisor, Biologist) \\
& \quad \circ (hasAdvisor^-)^* \circ (hasAdvisor^-, Physicist) \circ (hasAdvisor^-)^*\big](x,y) \\
q_2(x,y) = {} & [\big(hasAdvisor, CompScientist \wedge Logician\big)^*](RKowalski, x), \\
& [\big(hasAdvisor, CompScientist \wedge Logician\big)^*](FBaader, x) \\
& [wroteThesis \circ submittedTo \circ locatedIn](x,y) \\
q_3(x) = {} & [(hasAdvisor, ((CompScientist \wedge Logician) \vee Physicist))^* \\
& \quad \circ (hasAdvisor, Biologist \vee Physicist)](RKowalski, x) \\
& [\big(hasAdvisor, ((CompScientist \wedge Logician) \vee Physicist)\big)^* \circ (hasAdvisor)](FBaader, x)
\end{aligned}
$$

Fig. 3: Example queries

We now define the semantics of ℓ-C2RPQs. We say that a set $\mathcal{X} \subset X$ satisfies a formula $\varphi \in \mathcal{B}(X)$, written $\mathcal{X} \models \varphi$, if the formula that results from replacing each $v \in X$ by $\mathbf{true}$ if $v \in \mathcal{X}$ and by $\mathbf{false}$ otherwise is equivalent to $\mathbf{true}$. For a regular language L over the alphabet $\mathcal{B}(\overline{\mathsf{N_R}}) \times \mathcal{B}(\mathsf{N_C})$, we call d_2 an L-successor of d_1 in $\mathcal{I}$ if there is some $w = (\Gamma_1, \Upsilon_1) \dots (\Gamma_n, \Upsilon_n) \in L$ and some sequence $e_0, \dots, e_n$ of elements in $\Delta^{\mathcal{I}}$ such that $e_0 = d_1$, $e_n = d_2$, and, for all $1 \leq i \leq n$:

$$\{R \in \overline{\mathsf{N_R}} \mid \langle e_{i-1}, e_i \rangle \in R^{\mathcal{I}}\} \models \Gamma_i \quad \text{and} \quad \{A \in \mathsf{N_C} \mid e_i \in A^{\mathcal{I}}\} \models \Upsilon_i.$$

A *match* for a Boolean ℓ-C2RPQ q in an interpretation $\mathcal{I}$ is a mapping π from the terms in q to elements in $\Delta^{\mathcal{I}}$ such that:

- $\pi(c) = c^{\mathcal{I}}$ if $c \in \mathsf{N_I}$,
- $\{A \in \mathsf{N_C} \mid \pi(t) \in A^{\mathcal{I}}\} \models \beta$ for each atom $\beta(t)$ in q, and
- $\pi(t')$ is an L-successor of $\pi(t)$ for each atom $L(t, t')$ in q.

We write $\mathcal{I} \models q$ if there is a match for q in $\mathcal{I}$, and $\mathcal{T}, \mathcal{A} \models q$ if $\mathcal{I} \models q$ for every model $\mathcal{I}$ of $\mathcal{T}, \mathcal{A}$.

Given an ℓ-C2RPQ q with answer variables $v_1, \ldots, v_k$, we say that a tuple of individuals $(a_1, \ldots, a_k)$ is a *certain answer* for q w.r.t. $\mathcal{T}, \mathcal{A}$ just in the case that in every model $\mathcal{I}$ of $\mathcal{T}, \mathcal{A}$ there is a match π for q such that $\pi(v_i) = a_i^{\mathcal{I}}$ for every $1 \leq i \leq k$. Just as for CQs and C2RPQs, deciding whether a tuple of individuals is a certain answer for an ℓ-C2RPQ can be linearly reduced to Boolean ℓ-C2RPQ entailment. For this reason, we consider only the latter problem in what follows.

It is well known that the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ can be homomorphically embedded into any model of $\mathcal{T}, \mathcal{A}$, hence a CQ q is entailed by $\mathcal{T}, \mathcal{A}$ if and only if there is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. This result can be easily lifted from CQs to ℓ-C2RPQs, as ℓ-C2RPQs are also monotonic and their matches are preserved under homomorphisms.

Lemma 1. *For every DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $(\mathcal{T}, \mathcal{A})$ and Boolean ℓ-C2RPQ q: $\mathcal{T}, \mathcal{A} \models q$ if and only if $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q$.*

This property will be a crucial element in establishing our main theorem:

Theorem 1. *Boolean ℓ-C2RPQ entailment is* NLSPACE*-complete in data complexity and* PSPACE*-complete in combined complexity for DL-Lite$_\mathcal{R}$; the combined complexity drops to* NP*-complete for DL-Lite$_{\mathsf{RDFS}}$. For $\mathcal{ELH}$, the problem is* P*-complete in data complexity and* PSPACE*-complete in combined complexity. All lower bounds hold also for CRPQs and in the absence of role inclusions.*

We split the proof of this theorem into parts, with the lower bounds shown in the next section, and the (more involved) proofs of the upper bounds outlined in Section 5.

4 Lower Bounds

We start by establishing the required lower bounds.

Proposition 1. *Boolean CRPQ entailment is*

1. NLSPACE*-hard in data complexity for DL-Lite$_{\mathsf{RDFS}}$;*
2. P*-hard in data complexity for $\mathcal{EL}$;*
3. NP*-hard in combined complexity for DL-Lite$_{\mathsf{RDFS}}$;*
4. PSPACE*-hard in combined complexity for DL-Lite and $\mathcal{EL}$.*

Proof. Statement (1) follows from the analogous result for graph databases [8]. It can be shown by a simple reduction from the NLSPACE-complete directed reachability problem: y is reachable from x in a directed graph G if and only if (x, y) is an answer to $r^*(x, y)$ w.r.t. the ABox $\mathcal{A}_G$ encoding G. Statement (2) is immediate given the P-hardness in data complexity of CQ entailment in $\mathcal{EL}$ [7], and (3) follows from the well-known NP-hardness in combined complexity of CQ entailment for databases [1].

For statement (4), we give a reduction from the problem of emptiness of the intersection of an arbitrary number of regular languages, which is known to PSPACE-complete [10]. Consider some regular languages $L_1, \ldots, L_n$ over alphabet Σ. We will use the symbols in Σ as role names, and we add a concept name A. Then we set $\mathcal{A} = \{A(a)\}$ and $q = \exists x\ L_1(a, x) \wedge \ldots \wedge L_n(a, x)$. For DL-Lite, we will use the

following TBox: $\mathcal{T} = \{A \sqsubseteq \exists r \mid r \in \Sigma\} \cup \{\exists r^- \sqsubseteq \exists s \mid r, s \in \Sigma\}$. For $\mathcal{EL}$, we can use $\mathcal{T} = \{A \sqsubseteq \exists r.A \mid r \in \Sigma\}$. Notice that in both cases the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ consists of an infinite tree rooted at a such that every element in the interpretation has a unique r-child for each $r \in \Sigma$ (and no other children). Thus, we can associate to every domain element the word over Σ given by the unique path from a, and moreover, for every word $w \in \Sigma^*$ we can find an element e_w whose path from a is exactly w. This means that if $w \in L_1 \cap \ldots \cap L_n$, we obtain a match for q in the canonical model by mapping x to e_w. Conversely, if q is entailed, then any match in the canonical model defines a word which belongs to every L_i, which means $L_1 \cap \ldots \cap L_n$ is non-empty. □

5 Upper Bounds

The main objective of this section will be to define procedure for deciding $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q$ for a given KB $\mathcal{T}, \mathcal{A}$ and a given ℓ-C2RPQ q. The procedure comprises two main steps. First, we rewrite q into a set Q of ℓ-C2RPQs such that $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q$ if and only if $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q'$ for some $q' \in Q$. The advantage of the rewritten queries is that in order to decide whether $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q'$, we will only need to consider matches which map the variables to $\mathsf{Ind}(\mathcal{A})$. The second step evaluates the rewritten queries over the core part of the canonical model involving only $\mathsf{Ind}(\mathcal{A})$.

Preliminary Notions In order to more easily manipulate regular languages, it will prove convenient to use NFAs rather than regular expressions. Thus, in what follows, we assume all binary atoms take the form $\alpha(t, t')$, where α is an NFA over $\mathcal{B}(\overline{\mathsf{N_R}}) \times \mathcal{B}(\mathsf{N_C})$. Given $\alpha = \langle S, \Sigma, \delta, s_0, F\rangle$, we use $\alpha_{s,G}$ to denote the NFA $\langle S, \Sigma, \delta, s, G\rangle$, i.e. the NFA with the same states and transitions as α but with initial state s and final states G.

A key to defining our rewriting procedure will be to understand how an atom $L(t, t')$ can be satisfied in the anonymous part of the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. A subtlety arises from the fact that the path witnessing the satisfaction of an atom $L(t, t')$ may be quite complicated: it may move both up and down, passing by the same element multiple times, and possibly descending below t'. This will lead us to decompose an atom $L(t, t')$ into multiple "smaller" atoms corresponding to segments of the L-path which are situated wholly above or below an element. Importantly, we know that the canonical model displays a high degree of regularity, since whenever two elements p_1 and p_2 in the anonymous part end with the same concept (i.e. $\mathsf{Tail}(p_1) = \mathsf{Tail}(p_2)$), the submodels $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_{p_1}$ and $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_{p_2}$ are isomorphic. In particular, this means that if $\mathsf{Tail}(p_1) = \mathsf{Tail}(p_2)$, then p_1 is an L-successor of itself in the interpretation $\mathcal{I}_{\mathcal{A},\mathcal{T}}|_{p_1}$ just in the case that p_2 is an L-successor of itself in the interpretation $\mathcal{I}_{\mathcal{A},\mathcal{T}}|_{p_2}$.

We now wish to define a way of testing for a given TBox $\mathcal{T}$ and NFA α with states s, s' whether $\mathsf{Tail}(e) = C$ ensures that there is a loop from e back to itself, situated wholly within $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_e$, which takes α from state s to state s'. To this end, we construct a table Loop_α which contains for each pair s, s' of states in α, a subset of BC. If $\mathcal{T}$ is a DL-Lite$_\mathcal{R}$ TBox, then Loop_α is defined inductively using the following rules:

(1) for every $s \in S$, $\mathsf{Loop}_\alpha[s, s] = \mathsf{BC}$
(2) if $C \in \mathsf{Loop}_\alpha[s_1, s_2]$ and $C \in \mathsf{Loop}_\alpha[s_2, s_3]$, then $C \in \mathsf{Loop}_\alpha[s_1, s_3]$

(3) if $C \in \mathsf{BC}$, $\mathcal{T} \models C \sqsubseteq \exists R$, $\exists R^- \in \mathsf{Loop}_\alpha[s_2, s_3]$,
$(s_1, (\Gamma, \Upsilon), s_2) \in \delta$, $(s_3, (\Gamma', \Upsilon'), s_4) \in \delta$,
$\{U \in \overline{\mathsf{N_R}} \mid \mathcal{T} \models R \sqsubseteq U\} \models \Gamma$, $\{A \in \mathsf{N_C} \mid \mathcal{T} \models \exists R^- \sqsubseteq A\} \models \Upsilon$,
$\{U \in \overline{\mathsf{N_R}} \mid \mathcal{T} \models R^- \sqsubseteq U\} \models \Gamma'$, and $\{A \in \mathsf{N_C} \mid \mathcal{T} \models C \sqsubseteq A\} \models \Upsilon'$,
then $C \in \mathsf{Loop}_\alpha[s_1, s_4]$

For $\mathcal{ELH}$, we replace the third rule by:

(3') if $C \in \mathsf{BC}$, $\mathcal{T} \models C \sqsubseteq \exists r.D$, $D \in \mathsf{Loop}_\alpha[s_2, s_3]$,
$(s_1, (\Gamma, \Upsilon), s_2) \in \delta$, $(s_3, (\Gamma', \Upsilon'), s_4) \in \delta$,
$\{s \in \mathsf{N_R} \mid \mathcal{T} \models r \sqsubseteq s\} \models \Gamma$, $\{A \in \mathsf{N_C} \mid \mathcal{T} \models D \sqsubseteq A\} \models \Upsilon$,
$\{s^- \in \overline{\mathsf{N_R}} \mid \mathcal{T} \models r \sqsubseteq s\} \models \Gamma'$, and $\{A \in \mathsf{N_C} \mid \mathcal{T} \models C \sqsubseteq A\} \models \Upsilon'$,
then $C \in \mathsf{Loop}_\alpha[s_1, s_4]$

Note that the table Loop_α can be constructed in polynomial time in $|\mathcal{T}|$ and $|\alpha|$ since entailment of inclusions is polynomial for both DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$. The following lemma shows that Loop_α has the desired meaning:

Lemma 2. *For every element $p \in \Delta^{\mathcal{I}_{\mathcal{A},\mathcal{T}}} \setminus \mathsf{Ind}(\mathcal{A})$: $\mathsf{Tail}(p) \in \mathsf{Loop}_\alpha[s, s']$ if and only if p is an $L(\alpha_{s,s'})$-successor of itself in the interpretation $\mathcal{I}_{\mathcal{A},\mathcal{T}}|_p$.*

Query Rewriting Our aim is to rewrite our query in such a way that we do not need to map any variables to the anonymous part of the model. We draw our inspiration from a query rewriting procedure for Horn-$\mathcal{SHIQ}$ described in [9]. The main intuition is as follows. Suppose we have a match π for q which maps some variable y to the anonymous part, and no other variable is mapped below $\pi(y)$. Then we modify q so that it has essentially the same match except that variables mapped to $\pi(y)$ are now mapped to the (unique) parent of $\pi(y)$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. The delicate point is that we must "split" atoms of the form $\alpha(t, t')$ with $y \in \{t, t'\}$ into the parts which are satisfied in the subtree $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_{\pi(y)}$ (these have already been shown to hold, so can be dropped), and those which occur above $\pi(y)$, whose satisfaction still needs to be determined and thus must be incorporated into the new query. With each iteration of the rewriting procedure, we obtain a query which has a match which maps variables "closer" to the core of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, until eventually we find some query that has a match which maps all terms to $\mathsf{Ind}(\mathcal{A})$.

We now give a recursive non-deterministic query rewriting procedure which implements the above intuition.

PROCEDURE rewrite$(q, \mathcal{T})$

1. Choose either to output q or to continue.
2. Choose a non-empty set $\mathsf{Leaf} \subseteq \mathsf{vars}(q)$ and $y \in \mathsf{Leaf}$. Rename all variables in Leaf to y.
3. Choose some concept $C \in \mathsf{BC}$ such that $\{B \mid \mathcal{T} \models C \sqsubseteq B\} \models \varphi$ for every atom $\varphi(y)$. Drop all such atoms from q.
4. For each atom $\alpha(t, t')$ where $\alpha = \langle S, \Sigma, \delta, s, F\rangle$ is a NFA and $y \in \{t, t'\}$,
   - choose a sequence $s_1, \ldots s_n$ of distinct states from S such that $s_n \in F$,
   - replace the atom $\alpha(t, t')$ in q by the atoms $\alpha_{s,s_1}(t, y)$, $\alpha_{s_1,s_2}(y, y)$, $\ldots$, $\alpha_{s_{n-2},s_{n-1}}(y, y)$, $\alpha_{s_{n-1},s_n}(y, t')$.

5. Drop all atoms $\alpha_{s,s'}(y,y)$ such that $C \in \mathsf{Loop}_\alpha[s,s']$.
6. Choose some $D \in \mathsf{BC}$ and $R \in \overline{\mathsf{N_R}}$ such that:
   (a) if $\mathcal{T}$ is a DL-Lite$_\mathcal{R}$ TBox, then $C = \exists R^-$ and $\mathcal{T} \models D \sqsubseteq \exists R$.
   (a') if $\mathcal{T}$ is an $\mathcal{ELH}$ TBox, then $R \in \mathsf{N_R}$ and $\mathcal{T} \models D \sqsubseteq \exists R.C$.
   (b) for each atom of the form $\alpha(y,x)$ with $\alpha = \langle S, \Sigma, \delta, s, F\rangle$, there exist $s' \in S$ and $(\Gamma, \Upsilon) \in \Sigma$ with $(s,(\Gamma,\Upsilon),s') \in \delta$ and $\{U \in \overline{\mathsf{N_R}} \mid \mathcal{T} \models R^- \sqsubseteq U\} \models \Gamma$.
   (c) for each atom of the form $\alpha(x,y)$ with $\alpha = \langle S, \Sigma, \delta, s, F\rangle$, there exists $s'' \in S$, $s_f \in F$ and some $(\Omega, \Theta) \in \Sigma$ with $(s'',(\Omega,\Theta),s_f) \in \delta$ such that $\{U \in \overline{\mathsf{N_R}} \mid \mathcal{T} \models R \sqsubseteq U\} \models \Omega$ and $\{A \in \mathsf{N_C} \mid \mathcal{T} \models C \sqsubseteq A\} \models \Theta$.
   Note that both (b) and (c) apply to an atom of the form $\alpha(y,y)$.
7. Replace
   – each atom $\alpha(y,x)$ with $x \neq y$ by atoms $\alpha_{s',s_f}(y,x)$ and $\Upsilon(y)$
   – each atom $\alpha(x,y)$ with $x \neq y$ by $\alpha_{s,s''}(x,y)$
   – each atom $\alpha(y,y)$ by atoms $\alpha_{s',s''}(y,y)$ and $\Upsilon(y)$
   with $s, s', s'', s_f, \Upsilon$ as in Step 6.
8. If $D \in \mathsf{N_C}$ is the concept chosen in Step 6, add $D(y)$ to q. If $D = \exists P^-$, add $\alpha_P(z,y)$ to q, where z is a fresh variable and $L(\alpha_P) = \{(P, \mathbf{true})\}$. Go to Step 1.

Slightly abusing notation, we will use $\mathsf{rewrite}(q,\mathcal{T})$ to denote the set of queries which are output by some execution of rewrite on input q,$\mathcal{T}$. We remark that the number of variables and atoms in each query in $\mathsf{rewrite}(q,\mathcal{T})$ is linearly bounded by the original q. This is the key property used to show the following:

Lemma 3. *There are only exponentially many queries in* $\mathsf{rewrite}(q,\mathcal{T})$ *(up to equivalence), and each one has size polynomial in* $|q|$.

The next lemma shows that using $\mathsf{rewrite}(q,\mathcal{T})$, we can reduce the problem of finding an arbitrary query match to finding a match involving only ABox individuals.

Lemma 4. $\mathcal{T}, \mathcal{A} \models q$ *if and only if there exists a match* π *for some query* $q' \in \mathsf{rewrite}(q,\mathcal{T})$ *in* $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ *such that* $\pi(t) \in \mathsf{Ind}(\mathcal{A})$ *for every term* t *in* q'.

Query Evaluation We have reduced $\mathcal{T}, \mathcal{A} \models q$ to checking whether there is a match π for some $q' \in \mathsf{rewrite}(q,\mathcal{T})$ with $\pi(t) \in \mathsf{Ind}(\mathcal{A})$ for every term t in q'. However, even when all terms are mapped to ABox individuals, the paths between them may need to pass by the anonymous part in order to satisfy the regular expressions in the query. To handle this problem, we define a relaxed notion of query entailment, which exploits the fact that if all variables are mapped to $\mathsf{Ind}(\mathcal{A})$, only loops (that is, paths from an individual a to itself in $\mathcal{I}_{\mathcal{A},\mathcal{T}}|_a$) may participate in the paths between them. Hence, we look for paths in the ABox that may use such loops to skip states in the query automata.

Thus, as part of our evaluation procedure, we will need to decide for a given individual a whether a is an $L(\alpha_{s,s'})$-successor of itself in $\mathcal{I}_{\mathcal{A},\mathcal{T}}|_a$. We cannot use the table Loop_α directly, since it does not take into account the concepts which are entailed due to ABox assertions. We note however that the set of loops starting from a given individual is fully determined by the set of concepts in BC which the individual satisfies. We thus introduce a new table ALoop_α such that $\mathsf{ALoop}_\alpha[s,s']$ contains all *subsets* $G \subseteq \mathsf{BC}$ such that a is an $L(\alpha_{s,s'})$-successor of itself in $\mathcal{I}_{\mathcal{A},\mathcal{T}}|_a$ whenever $G = \{C \in \mathsf{BC} \mid a \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}\}$. Note that the size of ALoop_α is exponential in $|\mathcal{T}|$, but the associated decision problem is in P:

Lemma 5. *It can be decided in polytime in $|\mathcal{T}|$ and $|\alpha|$ whether $G \in \mathsf{ALoop}_\alpha[s, s']$.*

Definition 2. *We write $\mathcal{T}, \mathcal{A} \mid\!\approx q$ if there is a mapping π from the terms in q to $\mathsf{Ind}(\mathcal{A})$ such that:*

(a) $\pi(c) = c$ *for each* $c \in \mathsf{N_I}$,
(b) $\{A \in \mathsf{N_C} \mid \mathcal{T}, \mathcal{A} \models A(\pi(t))\} \models \beta$ *for each atom* $\beta(t)$ *in* q, *and*
(c) for each $\alpha(t, t') \in q$ *with* $\alpha = \langle S, \Sigma, \delta, s, F\rangle$, *there is a sequence* $(a_0, s_0), \dots (a_n, s_n)$ *of distinct pairs from* $\mathsf{Ind}(\mathcal{A}) \times S$ *such that* $a_0 = \pi(t)$, $a_n = \pi(t')$, $s_0 = s$, $s_n \in F$, *and for every* $0 \leq i < n$, *one of the following holds:*
  (i) $a_i = a_{i+1}$ *and* $\{C \in \mathsf{BC} \mid \mathcal{T}, \mathcal{A} \models C(a_i)\} \in \mathsf{ALoop}_\alpha[s_i, s_{i+1}]$
  (ii) there is some $(\Gamma, \Upsilon) \in \Sigma$ *with* $(s_i, (\Gamma, \Upsilon), s_{i+1})$, $\{R \in \overline{\mathsf{N_R}} \mid \mathcal{T}, \mathcal{A} \models R(a_i, a_{i+1})\} \models \Gamma$ *and* $\{A \in \mathsf{N_C} \mid \mathcal{T}, \mathcal{A} \models A(a_{i+1})\} \models \Upsilon$.

Lemma 6. $\mathcal{T}, \mathcal{A} \models q$ *if and only if* $\mathcal{T}, \mathcal{A} \mid\!\approx q'$ *for some* $q' \in \mathsf{rewrite}(q, \mathcal{T})$.

Using the preceding lemma, we can derive our upper bounds:

Proposition 2. *Boolean ℓ-C2RPQ entailment is*

1. NLSPACE *in data complexity for DL-Lite$_\mathcal{R}$ and DL-Lite$_\mathsf{RDFS}$;*
2. P *in data complexity for $\mathcal{ELH}$;*
3. NP *in combined complexity for DL-Lite$_\mathsf{RDFS}$;*
4. PSPACE *in combined complexity for DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$.*

Proof. By Lemmas 4 and 6, we can reduce $\mathcal{T}, \mathcal{A} \models q$ to deciding whether $\mathcal{T}, \mathcal{A} \mid\!\approx q'$ for some $q' \in \mathsf{rewrite}(q, \mathcal{T})$. For items 1 and 2, if $\mathcal{T}$ and q are fixed, then computing $\mathsf{rewrite}(q, \mathcal{T})$ requires only constant time in $|\mathcal{A}|$. To decide whether $\mathcal{T}, \mathcal{A} \mid\!\approx q'$ for $q' \in \mathsf{rewrite}(q, \mathcal{T})$, we guess a mapping π from the terms in q' to $\mathsf{Ind}(\mathcal{A})$ and verify that it satisfies the conditions in Definition 2. Note that for condition (c), we cannot keep the whole sequence $(a_0, s_0), \dots (a_n, s_n)$ in memory at once, so we use a binary counter that counts up to $\mathsf{Ind}(\mathcal{A}) \times |S|$ and store only one pair of nodes $(a_i, s_i), (a_{i+1}, s_{i+1})$ at a time. To verify conditions (b) and (c)(ii) we need checks of the form $\{R \in \overline{\mathsf{N_R}} \mid \mathcal{T}, \mathcal{A} \models R(a, b)\} \models \Gamma$ and $\{A \in \mathsf{N_C} \mid \mathcal{T}, \mathcal{A} \models A(a)\} \models \Upsilon$. Each one amounts to a fixed number of instance checks (one for each symbol in Γ or Υ), hence the data complexity of these checks is the same as for instance checking in the corresponding DL: in AC_0 for DL-Lite$_\mathcal{R}$, and in P for $\mathcal{ELH}$. This yields the desired upper bounds: NLSPACE for the former, and NLSPACE $^{\mathrm{P}}$ =P for the latter.

For statement 4, instead of building the whole set $\mathsf{rewrite}(q, \mathcal{T})$, which can be exponential, we generate a single $q' \in \mathsf{rewrite}(q, \mathcal{T})$ non-deterministically. More precisely, we take the initial query q and apply a sequence of rewriting steps to obtain some $q' \in \mathsf{rewrite}(q, \mathcal{T})$. By Lemma 3, every query in $\mathsf{rewrite}(q, \mathcal{T})$ can be generated after at most exponentially many steps, so we can use a polynomial-sized counter to check when we have reached this limit. Since each rewritten query is of polynomial size (Lemma 3), and we keep a single query in memory at a time, the generation of a single query in $\mathsf{rewrite}(q, \mathcal{T})$ requires only polynomial space. Then we can use the same strategy as above to decide in polynomial space whether $\mathcal{T}, \mathcal{A} \mid\!\approx q'$. We thus

have a non-deterministic polynomial space procedure for deciding $\mathcal{T}, \mathcal{A} \models q$. Using the well-known fact that NPSPACE =PSPACE, we obtain the desired upper bound.

For statement 3, we note that if $\mathcal{T}$ is an DL-Lite$_{\mathsf{RDFS}}$ TBox, then the query cannot be rewritten, i.e. $\mathsf{rewrite}(q, \mathcal{T}) = \{q\}$. Thus, it suffices to decide whether $\mathcal{T}, \mathcal{A} \not\models q$. We then remark that the procedure described above is in NP, since we guess a (polysize) mapping π and verify in polytime that π satisfies the conditions of Definition 2. □

6 Future Work

In future work, we plan to study what types of restrictions on the TBox and query lead to better combined complexity. We also wish to explore other types of path-based query languages. One interesting extension which has been recently proposed for graph databases [3] is the addition of *path variables*. In Boolean queries, these prove useful for speaking about equality of paths. For example, one might want to find whether there is a chain of advisors of the same length between both Edmund Clarke and Bernoulli, and Jack Minker and Bernoulli. Things become even more interesting if we allow path variables in the output. By outputting the paths for the preceding query, we could discover that Clarke and Minker have a path to Bernoulli of length 12. We could even output the common areas of expertise of the scientists along the path to find out that both have an 8-step path to some physicist (Poisson and Fourier), that in turn reach Bernoulli via an important figure in analysis (Lagrange) and a graph theorist (Euler).

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proc. of IJCAI. pp. 364–369 (2005)
3. Barceló, P., Hurtado, C.A., Libkin, L., Wood, P.T.: Expressive languages for path queries over graph-structured data. In: Proc. of PODS. pp. 3–14 (2010)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
5. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proc. of AAAI. pp. 391–396 (2007)
6. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Proc. of IJCAI. pp. 714–720 (2009)
7. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR. pp. 260–270 (2006)
8. Consens, M.P., Mendelzon, A.O.: Graphlog: a visual formalism for real life recursion. In: Proc. of PODS. pp. 404–416 (1990)
9. Eiter, T., Ortiz, M., Šimkus, M., Tran, T., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of AAAI (2012)
10. Kozen, D.: Lower bounds for natural proof systems. In: Proc. of FOCS. pp. 254–266 (1977)
11. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the horn fragments of the description logics SHOIQ and SROIQ. In: Proc. of IJCAI. pp. 1039–1044 (2011)