Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Conjunctive Regular Path Queries in Lightweight Description Logics

Meghyn Bienvenu
Laboratoire de Recherche en Informatique
CNRS & Université Paris Sud, France

Magdalena Ortiz and **Mantas Šimkus**
Institute of Information Systems
Vienna University of Technology, Austria

Abstract

Conjunctive regular path queries are an expressive extension of the well-known class of conjunctive queries and have been extensively studied in the database community. Somewhat surprisingly, there has been little work aimed at using such queries in the context of description logic (DL) knowledge bases, and all existing results target expressive DLs, even though lightweight DLs are considered better-suited for data-intensive applications. This paper aims to bridge this gap by providing algorithms and tight complexity bounds for answering two-way conjunctive regular path queries over DL knowledge bases formulated in lightweight DLs of the DL-Lite and $\mathcal{EL}$ families.

1 Introduction

Recent years have seen a rapidly growing interest in using description logic (DL) ontologies to query instance data. In databases, similar attention has been paid to the related problem of querying graph databases which, like DL instance data, are sets of ground facts using only unary and binary predicates, i.e., node- and edge-labeled graphs [Consens and Mendelzon, 1990; Barceló *et al.*, 2010]. The relevance of both problems lies in the fact that in many application areas, data can be naturally represented in such form. This applies, in particular, to XML and RDF data. While the DL and database communities share some common research goals, the research agendas they have pursued differ significantly. In DLs, the focus has been on designing efficient algorithms for answering (plain) conjunctive queries in the presence of 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. Such path navigation has long been considered crucial for querying data on the web. Indeed, it lies at the core of the XPath language for querying XML data, and of the *property paths* feature of SPARQL 1.1, the language recently recommended as the new standard for querying RDF data.

In this paper, we are interested in the problem of querying DL knowledge bases using various kinds of regular path queries. We mainly focus on *conjunctive (two-way) regular path queries* (C(2)RPQs), which are one of the most expressive and popular languages for querying graph databases. CRPQs 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 (or roles, in DL parlance). In the case of 2RPQs and C2RPQs, roles can be navigated in both directions. C2RPQs have already been studied for DLs, but all existing results concern expressive DLs for which reasoning is provably intractable. In particular, algorithms have been proposed for $\mathcal{ZIQ}$, $\mathcal{ZIO}$, and $\mathcal{ZOQ}$ [Calvanese *et al.*, 2007b; 2009], 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 in Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ were proposed [Ortiz *et al.*, 2011]. These algorithms run in polynomial time in the size of the data, but still require exponential time in the size of the ontology. By contrast, to the best of our knowledge, path queries have not yet been considered for the lightweight DLs of the DL-Lite [Calvanese *et al.*, 2007a] and $\mathcal{EL}$ [Baader *et al.*, 2005] families, which underly the OWL 2 QL and EL profiles. This is surprising given that the low complexity of these DLs makes them better suited for data-intensive applications.

This paper aims to remedy this situation by providing algorithms and precise complexity bounds for answering (C)2RPQs in the $\mathcal{EL}$ and DL-Lite families of lightweight DLs. We show that in data complexity, the query answering problem for CR2PQs is NL-complete for DL-Lite and P-complete for $\mathcal{EL}$, which in both cases is the lowest complexity that could be expected. For combined complexity, we prove PSPACE-completeness for both DL-Lite and $\mathcal{EL}$, but somewhat surprisingly obtain a tractability result for 2RPQs for both DLs. All of our upper bounds apply to extensions of $\mathcal{EL}$ and DL-Lite with role inclusions. Full proofs of all results can be found in a technical report [Bienvenu *et al.*, 2013].

2 Preliminaries

We briefly recall the syntax of DL-Lite$_{\mathcal{R}}$ [Calvanese *et al.*, 2007a] and $\mathcal{ELH}$ [Baader *et al.*, 2005] (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 $b, c \in \mathsf{N_I}$. We use $\mathsf{Ind}(\mathcal{A})$ to refer to the set of individuals in $\mathcal{A}$. 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 $r, s \in \mathsf{N_R}$. Note that in $\mathcal{EL}(\mathcal{H})$ TBoxes, inverse roles are not permitted. 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. $(\neg A)^\mathcal{I} = \Delta^\mathcal{I} \setminus A^\mathcal{I}$, $(\exists r.C)^\mathcal{I} = \{c \mid \exists d : (c, d) \in r^\mathcal{I}, d \in C^\mathcal{I}\}$, and $(P^-)^\mathcal{I} = \{(c, d) \mid (d, c) \in P^\mathcal{I}\}$. An interpretation $\mathcal{I}$ 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 $(\mathcal{T}, \mathcal{A})$ 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 in *normal form*, meaning that all concept inclusions are of one of the following forms:

$$A \sqsubseteq B \quad A_1 \sqcap A_2 \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.B \sqsubseteq A$$

with $A, A_1, A_2, B \in \mathsf{N_C} \cup \{\top\}$. It is well-known (cf. [Baader *et al.*, 2005] that for every $\mathcal{ELH}$ TBox $\mathcal{T}$, one can construct in polynomial time an $\mathcal{ELH}$ TBox $\mathcal{T}'$ in normal form (possibly using new concept names) such that $\mathcal{T}' \models \mathcal{T}$ and every model of $\mathcal{T}$ can be expanded to a model of $\mathcal{T}'$.

We use $\mathsf{sig}(\mathcal{T})$ to denote the set of all concept and role names appearing in a TBox $\mathcal{T}$. For ease of reference, we also define sets $\mathsf{BC}_\mathcal{T}$ of *basic concepts* and $\mathsf{TC}_\mathcal{T}$ of *tail concepts* for $\mathcal{T}$ as follows: $\mathsf{BC}_\mathcal{T} = \mathsf{TC}_\mathcal{T} = \mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})$ if $\mathcal{T}$ is an $\mathcal{ELH}$ TBox, and $\mathsf{TC}_\mathcal{T} = \{\exists r, \exists r^- \mid r \in \mathsf{N_R} \cap \mathsf{sig}(\mathcal{T})\}$ and $\mathsf{BC}_\mathcal{T} = (\mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})) \cup \mathsf{TC}_\mathcal{T}$ for a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$.

Canonical Models We recall the definition of canonical models for DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ KBs. For both DLs, the domain of the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ for a KB $(\mathcal{T}, \mathcal{A})$ consists of *paths* of the form $aR_1C_1 \ldots R_nC_n$ ($n \geq 0$), where $a \in \mathsf{Ind}(\mathcal{A})$, each C_i is a tail 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 $aR_1\exists R_1^- \ldots R_n\exists R_n^-$ 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 an $\mathcal{ELH}$ TBox in normal form, the domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ contains exactly those paths $ar_1A_1 \ldots r_nA_n$ 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:

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

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 $F \subseteq S$ 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 Path Queries

We now introduce the query languages studied in this paper.

Definition 1. A *conjunctive (two-way) regular path query (C2RPQ)* has the form $q(\vec{x}) = \exists \vec{y}\ \varphi$ where $\vec{x}$ and $\vec{y}$ are tuples of variables, and φ is a conjunction of atoms of the forms:

(i) $A(t)$, where $A \in \mathsf{N_C}$ and $t \in \mathsf{N_I} \cup \vec{x} \cup \vec{y}$, and

(ii) $L(t, t')$, where L is (an NFA or regular expression defining) a regular language over $\overline{\mathsf{N_R}} \cup \{A? \mid A \in \mathsf{N_C}\}$, and $t, t' \in \mathsf{N_I} \cup \vec{x} \cup \vec{y}$.

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

Conjunctive (one-way) regular path queries (CRPQs) are obtained by disallowing symbols from $\overline{\mathsf{N_R}} \setminus \mathsf{N_R}$ in atoms of type (ii), and *conjunctive queries (CQs)* result from only allowing type-(ii) atoms of the form $r(t, t')$ with $r \in \mathsf{N_R}$. *Two-way regular path queries (2RPQs)* consist of a single atom of type (ii), and *regular path queries (RPQs)* further disallow symbols from $\overline{\mathsf{N_R}} \setminus \mathsf{N_R}$. Finally, *instance queries (IQs)* take the form $A(x)$ with $A \in \mathsf{N_C}$, or $r(x, y)$ with $r \in \mathsf{N_R}$.

We now define the semantics of C2RPQs. For a regular language L over the alphabet $\mathsf{N_R} \cup \overline{\mathsf{N_R}} \cup \{A? \mid A \in \mathsf{N_C}\}$, we call d_2 an *L-successor* of d_1 in $\mathcal{I}$ if there is some $w = u_1 \ldots u_n \in L$ and some sequence $e_0, \ldots, e_n$ of elements in $\Delta^\mathcal{I}$ such that $e_0 = d_1$, $e_n = d_2$, and, for all $1 \leq i \leq n$:
- if $u_i = A?$, then $e_{i-1} = e_i \in A^\mathcal{I}$
- if $u_i = R \in \mathsf{N_R} \cup \overline{\mathsf{N_R}}$, then $\langle e_{i-1}, e_i\rangle \in R^\mathcal{I}$

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

	IQ		(2)RPQ		CQ		C(2)RPQ	
	data	combined	**data**	**combined**	data	combined	**data**	**combined**
DL-Lite$_{\mathsf{RDFS}}$	in AC$_0$	NL-c	**NL-c**	**NL-c**	in AC$_0$	NP-c	**NL-c**	**NP-c**
DL-Lite$_{(\mathcal{R})}$	in AC$_0$	NL-c	**NL-c**	**P-c**†	in AC$_0$	NP-c	**NL-c**	**PSPACE-c**
$\mathcal{EL}(\mathcal{H})$	P-c	P-c	**P-c**	**P-c**	P-c	NP-c	**P-c**	**PSPACE-c**

Figure 1: Complexity of Boolean query entailment. The 'c' indicates completeness results. New results are marked in bold. For existing results, we refer to [Baader *et al.*, 2005; Calvanese *et al.*, 2007a; Rosati, 2007; Krisnadhi and Lutz, ; Krötzsch and Rudolph, 2007; Artale *et al.*, 2009] and references therein. † P-hardness for RPQs applies only to DL-Lite$_{\mathcal{R}}$.

– $\pi(c) = c^{\mathcal{I}}$ if $c \in \mathsf{N_I}$,
– $\pi(t) \in A^{\mathcal{I}}$ for each atom $A(t)$ in q, and
– $\pi(t')$ is an L-successor of $\pi(t)$, for each $L(t, t')$ 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 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 $1 \leq i \leq k$. Deciding whether a tuple of individuals is a certain answer for an C2RPQ can be linearly reduced to Boolean 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 from $\mathcal{T}, \mathcal{A}$ 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 C2RPQs, as C2RPQs are also monotonic and their matches are preserved under homomorphisms.

Lemma 2. *For every DL-Lite$_{\mathcal{R}}$ or $\mathcal{ELH}$ KB $(\mathcal{T}, \mathcal{A})$ and Boolean 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 3. *The complexity results in Figure 1 hold.*

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 4. *Boolean CRPQ entailment is*

1. NL-*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 & $\mathcal{EL}$.*

Statements (1) and (2) hold even for RPQs.

Proof. Statement (1) follows from the analogous result for graph databases [Consens and Mendelzon, 1990]. It can be shown by a simple reduction from the NL-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 instance checking in $\mathcal{EL}$ [Calvanese *et al.*, 2006], and (3) follows from the well-known NP-hardness in combined complexity of CQ entailment for databases [Abiteboul *et al.*, 1995].

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 be PSPACE-complete [Kozen, 1977]. Let $L_1, \ldots, L_n$ be regular languages over alphabet Σ. We will use the symbols in Σ as role names, and we add a concept name A. Let $\mathcal{A} = \{A(a)\}$ 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. ☐

For (2)RPQs in DL-Lite$_{\mathsf{RDFS}}$ and $\mathcal{EL}$, we inherit combined complexity lower bounds of NL and P respectively from IQs. For DL-Lite, we establish a P lower bound for 2RPQs, which contrasts with the NL-completeness of instance checking.

Proposition 5. *Boolean 2RPQ entailment in DL-Lite is P-hard in combined complexity, assuming an NFA representation of the regular language.*

Proof sketch. Consider the P-complete entailment problem in which one is given a propositional formula $T = \rho_1 \wedge \ldots \wedge \rho_m \wedge v_1$ over variables $v_1, \ldots, v_n$ with $\rho_i = v_{i_1} \wedge v_{i_2} \rightarrow v_{i_3}$, and the problem is to decide whether $T \models v_n$. We construct a DL-Lite TBox $\mathcal{T}$ and 2RPQ q such that $\mathcal{T}, \{A(a)\} \models q$ if and only if $T \models v_n$. We let $\mathcal{T}$ consist of the axioms:

– $A \sqsubseteq \exists r_{i,j}$, for $1 \leq i \leq m$, $j \in \{1, 2\}$
– $\exists r^-_{i_1,j_1} \sqsubseteq \exists r_{i_2,j_2}$, for $1 \leq i_1, i_2 \leq m$ and $j_1, j_2 \in \{1, 2\}$

and $q = \exists x\, \alpha(x, x)$, where $\alpha = (S, \Sigma, \delta, s_0, \{v_n^{out}\})$ is the NFA defined as follows:

– $S = \{s_0\} \cup \{v_1\} \cup \{v_i^{in}, v_i^{out} \mid 2 \leq i \leq n\} \cup \{\rho_i \mid 1 \leq i \leq m\}$

– $\Sigma = \{A?\} \cup \{r_{i,j}, r_{i,j}^- \mid 1 \leq i \leq m, 1 \leq j \leq 2\}$

– δ contains $(s_0, A?, v_n^{in})$, and for each $\rho_i = v_j \wedge v_k \to v_\ell$, the following transitions: $(v_\ell^{in}, r_{i,1}, v_j^{in})$, $(v_j^{out}, r_{i,1}^-, \rho_i)$, $(\rho_i, r_{i,2}, v_k^{in})$, and $(v_k^{out}, r_{i,2}^-, v_\ell^{out})$. Note: we use v_1 in place of v_1^{in} and v_1^{out}.

The first transition in δ enforces that x must be mapped to a and that there must be a loop at a from state v_n^{in} to v_n^{out}. Intuitively, a state v_ℓ^{in} indicates that v_ℓ needs to be proven, and v_ℓ^{out} signals that v_ℓ has been successfully derived. From a state v_ℓ^{in}, the available transitions correspond to the rules in T which conclude on v_ℓ: selecting transition $(v_\ell^{in}, r_{i,1}, v_j^{in})$ means choosing to use ρ_i to derive v_ℓ. The transitions $(v_\ell^{out}, r_{i,1}^-, \rho_i)$ and $(\rho_i, r_{i,2}, v_k^{in})$ allow us to move to the second variable of ρ_i once the first variable of ρ_i has been derived. When both variables have been proven, the transition $(v_k^{out}, r_{i,2}^-, v_\ell^{out})$ allows us to exit the derivation of v_ℓ. Thus, any loop from v_n^{in} to v_n^{out} in $\mathcal{I}_{\mathcal{T},\{A(a)\}}$ corresponds to a derivation of v_n, and conversely, any derivation of v_n yields a path witnessing the entailment of q. $\square$

As a corollary, we get P-hardness of RPQs in DL-Lite$_\mathcal{R}$, by using role inclusions to simulate the inverse roles in the query. We leave open whether the preceding hardness result applies when regular languages are given as regular expressions.

5 Upper Bounds

The main objective of this section will be to define a procedure for deciding $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q$ for a KB $(\mathcal{T}, \mathcal{A})$ and C2RPQ q. The procedure comprises two main steps. First, we rewrite q into a set Q of 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 decides the existence of such restricted matches for the rewritten queries.

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 $\mathsf{N_R} \cup \overline{\mathsf{N_R}} \cup \{A? \mid A \in \mathsf{N_C}\}$. 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.

5.1 Loop Computation

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 will require 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 $\mathsf{TC}_\mathcal{T}$. If $\mathcal{T}$ is a DL-Lite$_\mathcal{R}$ TBox, then Loop_α is defined inductively using the following rules:

1. for every $s \in S$: $\mathsf{Loop}_\alpha[s, s] = \mathsf{TC}_\mathcal{T}$
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 $C \in \mathsf{TC}_\mathcal{T}$, $\mathcal{T} \models C \sqsubseteq A$, and $(s_1, A?, s_2) \in \delta$, then $C \in \mathsf{Loop}_\alpha[s_1, s_2]$
4. if $\mathcal{T} \models C \sqsubseteq \exists R$, $\mathcal{T} \models R \sqsubseteq R'$, $\mathcal{T} \models R^- \sqsubseteq R''$, $(s_1, R', s_2) \in \delta$, $\exists R^- \in \mathsf{Loop}_\alpha[s_2, s_3]$, $(s_3, R'', s_4) \in \delta$, $C \in \mathsf{TC}_\mathcal{T}$, and $C \neq \exists R$, then $C \in \mathsf{Loop}_\alpha[s_1, s_4]$

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

4'. if $\mathcal{T} \models C \sqsubseteq \exists r.D$, $\mathcal{T} \models r \sqsubseteq r'$, $\mathcal{T} \models r \sqsubseteq r''$, $(s_1, r', s_2) \in \delta$, $D \in \mathsf{Loop}_\alpha[s_2, s_3]$, $(s_3, r''^-, s_4) \in \delta$, and $C \in \mathsf{TC}_\mathcal{T}$, then $C \in \mathsf{Loop}_\alpha[s_1, s_4]$

Example 6. Consider a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$ containing the inclusions $B \sqsubseteq \exists r$, $\exists r^- \sqsubseteq B$, $B \sqsubseteq \exists t_1$, and $t_1 \sqsubseteq t_2^-$, and consider the query $q = \exists xy.r^* t_1 t_2 r^-(x, y), B(y)$, or equivalently, $q = \exists xy.\alpha(x, y), B(y)$, where $\alpha = \langle\{s_0, s_1, s_2, s_3\}, \{r, t_1, t_2, r^-\}, \delta, s_0, \{s_3\}\rangle$ and $\delta = \{(s_0, r, s_0), (s_0, t_1, s_1), (s_1, t_2, s_2), (s_2, r^-, s_3)\}$. In the first step of the loop computation, we infer that $\mathsf{Loop}_\alpha[s_i, s_i]$ is the set of all tail concepts for $0 \leq i \leq 4$. Next, by rule 4, and using $\mathcal{T} \models B \sqsubseteq \exists t_1$, $\mathcal{T} \models \exists r^- \sqsubseteq \exists t_1$, $\mathcal{T} \models t_1^- \sqsubseteq t_2$, (s_0, t_1, s_1), $\exists t_1^- \in \mathsf{Loop}_\alpha[s_1, s_1]$, and (s_1, t_2, s_2), we can infer that $B \in \mathsf{Loop}_\alpha[s_0, s_2]$ and $\exists r^- \in \mathsf{Loop}_\alpha[s_0, s_2]$. In a further step, we can use $\mathcal{T} \models B \sqsubseteq \exists r$, (s_0, r, s_0), $\exists r^- \in \mathsf{Loop}_\alpha[s_0, s_2]$, and (s_2, r^-, s_3) to obtain $B \in \mathsf{Loop}_\alpha[s_0, s_3]$.

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 7. *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 $\mathcal{I}_{\mathcal{A},\mathcal{T}}|_p$.*

5.2 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 [Eiter *et al.*, 2012]. 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"

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 $C \in \mathsf{TC}_{\mathcal{T}}$ such that $\mathcal{T} \models C \sqsubseteq B$ whenever $B(y)$ is an atom of q. Drop all such atoms from q.
4. For each atom $\alpha(t, t')$ where $\alpha = \langle S, \Sigma, \delta, s, F\rangle$ is an 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 $\alpha(t, t')$ 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 $D \in \mathsf{BC}_{\mathcal{T}}$ and $R, R_1, R_2 \in \overline{\mathsf{N_R}}$ such that:
   (a) $C = \exists R^-$ and $\mathcal{T} \models D \sqsubseteq \exists R$ [for DL-Lite$_R$], or $R \in \mathsf{N_R}$ and $\mathcal{T} \models D \sqsubseteq \exists R.C$ [for $\mathcal{ELH}$].
   (b) $\mathcal{T} \models R \sqsubseteq R_1$ and $\mathcal{T} \models R \sqsubseteq R_2$
   (c) for each atom $\alpha(y, x)$ with $\alpha = \langle S, \Sigma, \delta, s, F\rangle$, there exists $s' \in S$ such that $(s, R_1^-, s') \in \delta$.
   (d) for each atom $\alpha(x, y)$ with $\alpha = \langle S, \Sigma, \delta, s, F\rangle$, there exist $s'' \in S$, $s_f \in F$ with $(s'', R_2, s_f) \in \delta$.

   For atoms of the form $\alpha(y, y)$, both (c) and (d) apply.
7. Replace
   - each atom $\alpha(y, x)$ with $x \neq y$ by $\alpha_{s',F}(y, x)$
   - each atom $\alpha(x, y)$ with $x \neq y$ by $\alpha_{s,s''}(x, y)$
   - each atom $\alpha(y, y)$ by atoms $\alpha_{s',s''}(y, y)$

   with s, s', s'', F 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 $L(\alpha_P) = \{P\}$. Go to Step 1.

Figure 2: Query rewriting procedure rewrite.

atoms of the form $\alpha(t, t')$ with $y \in \{t, t'\}$ into the parts which are satisfied in the subtree $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_{\pi(y)}$, 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 $\mathsf{Ind}(\mathcal{A})$.

In Figure 2, we give a recursive non-deterministic query rewriting procedure rewrite which implements the above intuition. Slightly abusing notation, we will use rewrite($q, \mathcal{T}$) to denote the set of queries which are output by some execution of rewrite on input q,$\mathcal{T}$.

Example 8. We illustrate two different ways to apply the rewriting step to the query $q = \exists xy.r^*t_1t_2r^-(x, y)$, $B(y)$ in Example 6. First, let $\mathsf{Leaf} = \{x\}$ be the set chosen in step 2. There is no renaming to do, so we proceed to step 3 and choose $\exists r^-$. In step 4, we choose the sequence s_2, s_3, and replace $\alpha(x, y)$ by $\alpha_{s_0,s_2}(x, x), \alpha_{s_2,s_3}(x, y)$. Since $\exists r^- \in \mathsf{Loop}_\alpha[s_0, s_2]$, we drop the first atom and keep only $\alpha_{s_2,s_3}(x, y)$. In step 6, we can choose B for D, and r for the roles R, R_1, R_2. This ensures (a) and (b). For (c), we can take s_3 since $(s_2, r^-, s_3) \in \delta$. In step 7, we replace $\alpha_{s_2,s_3}(x, y)$ by $\alpha_{s_3,s_3}(x, y)$. At the end of step 8, we are left with the query $q' = \exists xyz.r(z, y), \alpha_{s_3,s_3}(x, y), B(y)$, which is output as a rewriting when we return to step 1. We remark that q' is equivalent modulo $\mathcal{T}$ to the simpler $\exists yz.r(z, y)$ since by choosing $x = y$, the atom $\alpha_{s_3,s_3}(x, y)$ is trivially satisfied, and $B(y)$ is enforced by $r(z, y)$ and the inclusion $\exists r^- \sqsubseteq B$. Intuitively, this rewriting captures the fact that, whenever we have an element e in an interpretation that satisfies $\exists r^-$, then we can map x to e, thereby ensuring that the initial segment $r^*t_1t_2$ is satisfied below e. Moreover, by mapping y to the r-predecessor of e, we satisfy the remaining r^-.

As further illustration, suppose that in step 2, we choose $\mathsf{Leaf} = \{x, y\}$, and let y be the selected variable. After renaming, we obtain $q = \exists y.\alpha(y, y)$, $B(y)$. In step 3, we choose $\exists r^-$, which leads us to drop the atom $B(y)$, leaving us with $\exists y.\alpha(y, y)$. In step 4, we choose the sequence that contains only s_3, so the atom $\alpha(y, y)$ is left untouched. Since $\exists r^- \in \mathsf{Loop}_\alpha[s_0, s_3]$, we can drop this atom in step 5, obtaining the empty query. In step 6, we choose $D = B$ and $R = R_1 = R_2 = r$. Step 7 is inapplicable since there are no binary atoms. Finally, in step 8, we add $B(y)$ to obtain the query $q = \exists y.B(y)$. Intuitively, this rewriting captures that if some element e satisfies B, then we can map both x and y to it to obtain a query match in which the regular expression $r^*t_1t_2r^-(x, y)$ is fully satisfied below e.

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

Lemma 9. *$\mathcal{T}, \mathcal{A} \models q$ if and only if there exists a match π for some query $q' \in$ 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'.*

We remark that the number of variables and atoms in each query in rewrite($q, \mathcal{T}$) is linearly bounded by $|q|$. This is the key property used to show the following:

Lemma 10. *There are only exponentially many queries in* rewrite($q, \mathcal{T}$) *(up to equivalence), each having size polynomial in $|q|$.*

5.3 Query Evaluation

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. This leads us to 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.

As part of our query 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 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 basic concepts which the individual satisfies. We thus define a new table ALoop_α such that $\mathsf{ALoop}_\alpha[s, s']$ contains all *subsets* $G \subseteq \mathsf{BC}_\mathcal{T}$ such that a is an $L(\alpha_{s,s'})$-successor of itself in $\mathcal{I}_{\mathcal{A},\mathcal{T}}|_a$ whenever $G = \{C \in \mathsf{BC}_\mathcal{T} \mid a \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}\}$. Note that the table ALoop_α is exponential in $|\mathcal{T}|$, but the associated decision problem is in P:

Lemma 11. *It can be decided in polytime in $|\mathcal{T}|$ and $|\alpha|$ whether $G \in \mathsf{ALoop}_\alpha[s, s']$.*

We use ALoop_α to define a relaxed notion of query match.

Definition 12. We write $\mathcal{T}, \mathcal{A} \models\!\approx q$ 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) $\mathcal{T}, \mathcal{A} \models A(\pi(t))$ for each atom $A(t)$ in q, and

(c) for each $\alpha(t, t') \in q$ with $\alpha = \langle S, \Sigma, \delta, s, F\rangle$, there is a sequence $(a_0, s_0), \ldots, (a_n, s_n)$ 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 $0 \leq i < n$, either:

  (i) $a_i = a_{i+1}$ and $\{C \in \mathsf{BC}_\mathcal{T} \mid \mathcal{T},\mathcal{A} \models C(a_i)\} \in \mathsf{ALoop}_\alpha[s_i, s_{i+1}]$, or

  (ii) $\mathcal{T}, \mathcal{A} \models R(a_i, a_{i+1})$ and $(s_i, R, s_{i+1}) \in \delta$ for some R.

The following lemma characterizes C2RPQ entailment in terms of relaxed matches.

Lemma 13. *$\mathcal{T}, \mathcal{A} \models q$ if and only if $\mathcal{T}, \mathcal{A} \models\!\approx q'$ for some $q' \in \mathsf{rewrite}(q, \mathcal{T})$.*

By applying the preceding characterization, we obtain our C2RPQ upper bounds:

Proposition 14. *Boolean C2RPQ entailment is*

1. NL *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 sketch. By Lemmas 9 and 13, $\mathcal{T}, \mathcal{A} \models q$ just in the case that $\mathcal{T}, \mathcal{A} \models\!\approx q'$ for some $q' \in \mathsf{rewrite}(q, \mathcal{T})$. For statements 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 $\mathcal{T}, \mathcal{A} \models\!\approx q'$ 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 12. Note that for condition (c), we cannot keep the whole sequence $(a_0, s_0), \ldots (a_n, s_n)$ 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. The data complexity of verifying conditions (b) and (c) is the same as for instance checking in the corresponding DL: AC_0 for DL-Lite$_\mathcal{R}$, and P for $\mathcal{ELH}$. This yields the desired upper bounds of NL and NL^P =P, respectively.

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. By Lemma 10, every query in $\mathsf{rewrite}(q, \mathcal{T})$ can be generated after at most exponentially many steps, so we can use a polynomial-size counter to check when we have reached this limit. Since each rewritten query is of polynomial size (Lemma 10), and we keep only one query in memory at a time, the generation of a single query in $\mathsf{rewrite}(q, \mathcal{T})$ requires only polynomial space. We can then use the same strategy as above to decide in polynomial space whether $\mathcal{T}, \mathcal{A} \models\!\approx q'$. This yields 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, $\mathsf{rewrite}(q, \mathcal{T}) = \{q\}$. Thus, it suffices to decide $\mathcal{T}, \mathcal{A} \models\!\approx q$, which can be done by guessing a mapping π and verifying in polytime that π satisfies the conditions of Definition 12. ☐

By moving to 2RPQs, we can achieve tractability even in combined complexity.

Proposition 15. *Boolean 2RPQ entailment is*

1. NL *in combined complexity for DL-Lite$_\mathsf{RDFS}$;*

2. P *in combined complexity for DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$.*

Proof sketch. For (1), we can iterate over all mappings π of the (at most two) query variables, and for each mapping, we check whether the conditions of Definition 12 are met using the same strategy as in the proof of point 1 of Proposition 14. Recall that in DL-Lite$_\mathsf{RDFS}$, instance checking is in NL w.r.t. combined complexity [Calvanese *et al.*, 2007a].

For (2), we first give a polynomial reduction to the problem of deciding whether $\mathcal{T}, \mathcal{A} \models\!\approx q'$ with q' a 2RPQ. When $q = \exists xy\, L(x, y)$ with $x \neq y$, we can simply replace q by $q' = \exists xy\ \Sigma^* \cdot L \cdot \Sigma^*(x, y)$, where $\Sigma = \mathsf{N_R} \cup \overline{\mathsf{N_R}} \cup \{A? \mid A \in \mathsf{N_C}\}$, since $\mathcal{T}, \mathcal{A} \models q$ iff $\mathcal{T}, \mathcal{A} \models\!\approx q'$. For queries of the form $\exists x L(x, x)$, the proof is more involved and passes by the definition of an alternative rewriting procedure for 2RPQs, which is similar in spirit to rewrite but is guaranteed to run in polynomial time. We can then check for a match of a 2RPQ in the ABox using essentially the same strategy as for (1), except that we must now perform some polynomial-time ALoop_α tests to verify condition (c) of Definition 12. ☐

6 Conclusion and Future Work

In this paper, we established tight complexity bounds for answering various forms of regular path queries over knowledge bases formulated in lightweight DLs from the DL-Lite and $\mathcal{EL}$ families. Our results demonstrate that the query answering problem for these richer query languages is often not much harder than for the CQs and IQs typically considered. Indeed, query answering remains tractable in data complexity for the highly expressive class of C2RPQs, and for 2RPQs, we even retain polynomial combined complexity.

In future work, we plan to explore other useful extensions of regular path queries, such as *nested path expressions* (along the lines of [Pérez *et al.*, 2010]), and the addition of *path variables* (recently explored in [Barceló *et al.*, 2010]).

Acknowledgements The authors were supported by a Université Paris-Sud Attractivité grant and the ANR project PAGODA ANR-12-JS02-007-01 (Bienvenu), the FWF project T515-N23 (Ortiz), and the FWF project P25518-N23 and the WWTF project ICT12-015 (Šimkus).

References

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Artale *et al.*, 2009] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research*, 36:1–69, 2009.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, pages 364–369, 2005.

[Barceló *et al.*, 2010] Pablo Barceló, Carlos A. Hurtado, Leonid Libkin, and Peter T. Wood. Expressive languages for path queries over graph-structured data. In *Proc. of PODS*, pages 3–14, 2010.

[Bienvenu *et al.*, 2013] Meghyn Bienvenu, Magdalena Ortiz, and Mantas Šimkus. Conjunctive Regular Path Queries in Lightweight Description Logics. Technical Report INFSYS RR-1843-13-01, Institute of Information Systems, Vienna University of Technology. Available at http://www.kr.tuwien.ac.at/research/reports/.

[Calvanese *et al.*, 2006] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proc. of KR*, pages 260–270, 2006.

[Calvanese *et al.*, 2007a] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[Calvanese *et al.*, 2007b] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proc. of AAAI*, pages 391–396, 2007.

[Calvanese *et al.*, 2009] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Regular path queries in expressive description logics with nominals. In *Proc. of IJCAI*, pages 714–720, 2009.

[Consens and Mendelzon, 1990] Mariano P. Consens and Alberto O. Mendelzon. Graphlog: a visual formalism for real life recursion. In *Proc. of PODS*, pages 404–416, 1990.

[Eiter *et al.*, 2012] Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, TrungKien Tran, and Guohui Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proc. of AAAI*, 2012.

[Kozen, 1977] Dexter Kozen. Lower bounds for natural proof systems. In *Proc. of FOCS*, pages 254–266, 1977.

[Krisnadhi and Lutz,] Adila Krisnadhi and Carsten Lutz. Data complexity in the $\mathcal{EL}$ family of description logics. In *Proc. of LPAR*, pages 333–347.

[Krötzsch and Rudolph, 2007] Markus Krötzsch and Sebastian Rudolph. Conjunctive queries for $\mathcal{EL}$ with composition of roles. In *Proc. of DL*, 2007.

[Ortiz *et al.*, 2011] Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In *Proc. of IJCAI*, pages 1039–1044, 2011.

[Pérez *et al.*, 2010] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. nSPARQL: A navigational language for RDF. *Journal of Web Semantics*, 8(4):255–270, 2010.

[Rosati, 2007] Riccardo Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proc. of DL*, 2007.