Nested Regular Path Queries in Description Logics

Meghyn Bienvenu
Lab. de Recherche en Informatique
CNRS & Univ. Paris Sud, France

Diego Calvanese
KRDB Research Centre
Free Univ. of Bozen-Bolzano, Italy

Magdalena Ortiz
Mantas Šimkus
Institute of Information Systems
Vienna Univ. of Technology, Austria

Abstract

Two-way regular path queries (2RPQs) have received increased attention recently due to their ability to relate pairs of objects by flexibly navigating graph-structured data. They are present in property paths in SPARQL 1.1, the new standard RDF query language, and in the XML query language XPath. In line with XPath, we consider the extension of 2RPQs with nesting, which allows one to require that objects along a path satisfy complex conditions, in turn expressed through (nested) 2RPQs. We study the computational complexity of answering nested 2RPQs and conjunctions thereof (CN2RPQs) in the presence of domain knowledge expressed in description logics (DLs). We establish tight complexity bounds in data and combined complexity for a variety of DLs, ranging from lightweight DLs (*DL-Lite*, $\mathcal{EL}$) up to highly expressive ones. Interestingly, we are able to show that adding nesting to (C)2RPQs does not affect worst-case data complexity of query answering for any of the considered DLs. However, in the case of lightweight DLs, adding nesting to 2RPQs leads to a surprising jump in combined complexity, from P-complete to EXP-complete.

1 Introduction

Both in knowledge representation and in databases, there has been great interest recently in expressive mechanisms for querying data, while taking into account complex domain knowledge (Calvanese, De Giacomo, and Lenzerini 2008; Glimm et al. 2008). Description Logics (DLs) (Baader et al. 2003), which on the one hand underlie the W3C standard Web Ontology Language (OWL), and on the other hand are able to capture at the intensional level conceptual modeling formalisms like UML and ER, are considered particularly well suited for representing a domain of interest (Borgida and Brachman 2003). In DLs, instance data, stored in a so-called ABox, is constituted by ground facts over unary and binary predicates (*concepts* and *roles*, respectively), and hence resembles data stored in graph databases (Consens and Mendelzon 1990; Barceló et al. 2012). There is a crucial difference, however, between answering queries over graph databases and over DL ABoxes. In the former, the data is assumed to be complete, hence query answering amounts to the standard database task of query evaluation. In the latter, it is typically assumed that the data is incomplete and additional domain knowledge is provided by the DL ontology (or TBox). Hence query answering amounts to the more complex task of computing *certain answers*, i.e., those answers that are obtained from all databases that both contain the explicit facts in the ABox and satisfy the TBox constraints. This difference has driven research in different directions.

In databases, expressive query languages for querying graph-structured data have been studied, which are based on the requirement of relating objects by flexibly navigating the data. The main querying mechanism that has been considered for this purpose is that of one-way and two-way regular path queries (RPQs and 2RPQs) (Cruz, Mendelzon, and Wood 1987; Calvanese et al. 2003), which are queries returning pairs of objects related by a *path* whose sequence of edge labels belongs to a regular language over the (binary) database relations and their inverses. Conjunctive 2RPQs (C2RPQs) (Calvanese et al. 2000) are a significant extension of such queries that add to the navigational ability the possibility of expressing arbitrary selections, projections, and joins over objects related by 2RPQs, in line with conjunctive queries (CQs) over relational databases. Two-way RPQs are present in the property paths in SPARQL 1.1 (Harris and Seaborne 2013), the new standard RDF query language, and in the XML query language XPath (Berglund and others 2010). An additional construct that is present in XPath is the possibility of using *existential test operators*, also known as *nesting*, to express sophisticated conditions along navigation paths. When an existential test $\langle E \rangle$ is used in a 2RPQ E', there will be objects along the main navigation path for E' that match positions of E' where $\langle E \rangle$ appears; such objects are required to be the origin of a path conforming to the (nested) 2RPQ E. It is important to notice that existential tests in general cannot be captured even by C2RPQs, e.g., when tests appear within a transitive closure of an RPQ. Hence, adding nesting effectively increases the expressive power of 2RPQs and of C2RPQs.

In the DL community, query answering has been investigated extensively for a wide range of DLs, with much of the work devoted to CQs. With regards to the complexity of query answering, attention has been paid on the one hand to *combined complexity*, i.e., the complexity measured considering as input both the query and the DL knowledge base (constituted by TBox and ABox), and on the other

hand to *data complexity*, i.e., when only the ABox is considered as input. For expressive DLs that extend $\mathcal{ALC}$, CQ answering is typically coNP-complete in data-complexity (Ortiz, Calvanese, and Eiter 2008), and 2EXP-complete in combined complexity (Glimm et al. 2008; Lutz 2008; Eiter et al. 2009). For lightweight DLs, instead, CQ answering is in AC^0 in data complexity for *DL-Lite* (Calvanese et al. 2007), and P-complete for $\mathcal{EL}$ (Krisnadhi and Lutz 2007). For both logics, the combined complexity is dominated by the NP-completeness of CQ evaluation over plain relational databases. There has also been some work on (2)RPQs and C(2)RPQs. For the very expressive DLs $\mathcal{ZIQ}$, $\mathcal{ZOQ}$, and $\mathcal{ZOI}$, where regular expressions over roles are present also in the DL, a 2EXP upper bound has been shown via techniques based on alternating automata over infinite trees (Calvanese, Eiter, and Ortiz 2009). For the Horn fragments of $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, P-completeness in data complexity and EXP/2EXP-completeness in combined complexity are known (Ortiz, Rudolph, and Simkus 2011). For lightweight DLs, tight bounds for answering 2RPQs and C2RPQs have only very recently been established by Bienvenu, Ortiz, and Simkus (2013): for (C)(2)RPQs, data complexity is NL-complete in *DL-Lite* and *DL-Lite*$_\mathcal{R}$, and P-complete in $\mathcal{EL}$ and $\mathcal{ELH}$. For all of these logics, combined complexity is P-complete for (2)RPQs and PSPACE-complete for C(2)RPQs.

Motivated by the expressive power of nesting in XPath and SPARQL, in this paper we significantly advance these latter lines of research on query answering in DLs, and study the impact of adding nesting to 2RPQs and C2RPQs. We establish tight complexity bounds in data and combined complexity for a variety of DLs, ranging from lightweight DLs of the *DL-Lite* and $\mathcal{EL}$ families up to the highly expressive ones of the $\mathcal{SH}$ and $\mathcal{Z}$ families. Our results are summarized in Table 1. For DLs containing at least $\mathcal{ELI}$, we are able to encode away nesting, thus showing that the worst-case complexity of query answering is not affected by this construct. Instead, for lightweight DLs (starting already from *DL-Lite*!), we show that adding nesting to 2RPQs leads to a surprising jump in combined complexity, from P-complete to EXP-complete. We then develop a sophisticated rewriting-based technique that builds on (but significantly extends) the one proposed by Bienvenu, Ortiz, and Simkus (2013), which we use to prove that the problem remains in NL for *DL-Lite*. We thus show that adding nesting to (C)2RPQs does not affect worst-case data complexity of query answering for lightweight DLs.

For lack of space, some proofs have been relegated to the appendix of the long version (Bienvenu et al. 2014).

2 Preliminaries

We briefly recall the syntax and semantics of description logics (DLs). As usual, we assume countably infinite, mutually disjoint sets $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ of *concept names*, *role names*, and *individuals*. We typically use A for concept names, p for role names, and a, b for individuals. An *inverse role* takes the form p^- where $p \in \mathsf{N_R}$. We let $\mathsf{N_R^\pm} = \mathsf{N_R} \cup \{p^- \mid p \in \mathsf{N_R}\}$ and denote by r elements of $\mathsf{N_R^\pm}$.

A DL knowledge base (KB) consists of a *TBox* and an *ABox*, whose forms depend on the DL in question. In the DL $\mathcal{ELHI}_\bot$, a TBox is defined as a set of *(positive) role inclusions* of the form $r \sqsubseteq r'$ and *negative role inclusions* of the form $r \sqcap r' \sqsubseteq \bot$ with $r, r' \in \mathsf{N_R^\pm}$, and *concept inclusions* of the form $C \sqsubseteq D$, where C and D are *complex concepts* formed according to the following syntax:[1]

$$C ::= \top \mid \bot \mid A \mid \exists r.C \mid C \sqcap C$$

with $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R^\pm}$.

Some of our results refer specifically to the *lightweight DLs* that we define next. $\mathcal{ELHI}$ is the fragment of $\mathcal{ELHI}_\bot$ that has no $\bot$. $\mathcal{ELH}$ and $\mathcal{ELI}$ are obtained by additionally disallowing inverse roles and role inclusions, respectively. *DL-Lite*$_\mathcal{R}$ is also a fragment of $\mathcal{ELHI}_\bot$, in which concept inclusions can only take the forms $B_1 \sqsubseteq B_2$ and $B_1 \sqcap B_2 \sqsubseteq \bot$, for B_i a concept name or concept of the form $\exists r.\top$ with $r \in \mathsf{N_R^\pm}$. *DL-Lite* is the fragment of *DL-Lite*$_\mathcal{R}$ that disallows (positive and negative) role inclusions.

An *ABox* is a set of assertions of the form $C(a)$ or $r(a, b)$, where C is a complex concept, $r \in \mathsf{N_R^\pm}$, and $a, b \in \mathsf{N_I}$. We use $\mathsf{Ind}(\mathcal{A})$ to refer to the set of individuals in $\mathcal{A}$.

Semantics. The semantics of DL KBs 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 $p \in \mathsf{N_R}$ to $p^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.[2] The function $\cdot^\mathcal{I}$ can be straightforwardly extended to complex concepts and roles. In the case of $\mathcal{ELHI}_\bot$, this is done as follows: $\top^\mathcal{I} = \Delta^\mathcal{I}$, $\bot^\mathcal{I} = \emptyset$, $(p^-)^\mathcal{I} = \{(c, d) \mid (d, c) \in p^\mathcal{I}\}$, $(\exists r.C)^\mathcal{I} = \{c \mid \exists d : (c, d) \in r^\mathcal{I}, d \in C^\mathcal{I}\}$, and $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$. An interpretation $\mathcal{I}$ satisfies an inclusion $G \sqsubseteq H$ if $G^\mathcal{I} \subseteq H^\mathcal{I}$, and it satisfies an assertion $C(a)$ (resp., $r(a, b)$) if $a^\mathcal{I} \in A^\mathcal{I}$ (resp., $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$). A *model* of a KB $(\mathcal{T}, \mathcal{A})$ is an interpretation $\mathcal{I}$ which satisfies all inclusions in $\mathcal{T}$ and assertions in $\mathcal{A}$.

Complexity. In addition to P and (co)NP, our results refer to the complexity classes NL (non-deterministic logarithmic space), PSPACE (polynomial space), and (2)EXP ((double) exponential time), cf. (Papadimitriou 1993).

3 Nested Regular Path Queries

We now introduce our query languages. In RPQs, nested RPQs and their extensions, *atoms* are given by (nested) regular expressions whose symbols are *roles*. The set Roles of roles contains $\mathsf{N_R^\pm}$, and all *test roles* of the forms $\{a\}?$ and $A?$ with $a \in \mathsf{N_I}$ and $A \in \mathsf{N_C}$. They are interpreted as $(\{a\}?)^\mathcal{I} = (a^\mathcal{I}, a^\mathcal{I})$, and $(A?)^\mathcal{I} = \{(o, o) \mid o \in A^\mathcal{I}\}$.

Definition 3.1. *A* nested regular expression *(NRE), denoted by E, is constructed according to the following syntax:*

$$E ::= \sigma \mid E \cdot E \mid E \cup E \mid E^* \mid \langle E \rangle$$

where $\sigma \in \mathsf{Roles}$.

[1] We slightly generalize the usual $\mathcal{ELHI}_\bot$ by allowing for negative role inclusions.

[2] Note that we do not make the *unique name assumption* (UNA), but all of our results continue to hold if the UNA is adopted.

	2RPQ		C2RPQ		N2RPQ / CN2RPQ	
	data	combined	data	combined	data	combined
Graph DBs & RDFS	NL-c	NL-c	NL-c	NP-c	NL-c	**P-c** / NP-c
DL-Lite	NL-c	P-c	NL-c	PSPACE-c	**NL-c**	**EXP-c**
Horn DLs (e.g., $\mathcal{EL}$, Horn-$\mathcal{SHIQ}$)	P-c	P-c	P-c	PSPACE-c	**P-c**	**EXP-c**
Expressive DLs (e.g., $\mathcal{ALC}$, $\mathcal{SHIQ}$)	coNP-h	EXP-c	coNP-h	2EXP-c	coNP-h	**EXP-c / 2EXP-c**

Table 1: Complexity of query answering. The 'c' indicates completeness, the 'h' hardness. New results are marked in bold. For existing results, refer to (Bienvenu, Ortiz, and Simkus 2013; Pérez, Arenas, and Gutierrez 2010; Barceló Baeza 2013; Calvanese, Eiter, and Ortiz 2009; Ortiz, Rudolph, and Simkus 2011) and references therein.

We assume a countably infinite set $\mathsf{N_V}$ *of* variables *(disjoint from* $\mathsf{N_C}$*,* $\mathsf{N_R}$*, and* $\mathsf{N_I}$*). Each* $t \in \mathsf{N_V} \cup \mathsf{N_I}$ *is a* term*. An* atom *is either a* concept atom *of the form* $A(t)$*, with* $A \in \mathsf{N_C}$ *and* t *a term, or a* role atom *of the form* $E(t, t')$*, with* E *an NRE and* t, t' *two (possibly equal) terms.*

A nested two-way regular path query *(N2RPQ)* $q(x, y)$ *is an atom of the form* $E(x, y)$*, where* E *is an NRE and* x, y *are two distinct variables. A* conjunctive N2RPQ *(CN2RPQ)* $q(\vec{x})$ *with* answer variables $\vec{x}$ *has the form* $\exists \vec{y}.\varphi$*, where* φ *is a conjunction of atoms whose variables are among* $\vec{x} \cup \vec{y}$*.*

A (plain) regular expression (RE) is an NRE that does not have subexpressions of the form $\langle E \rangle$. Two-way regular path queries *(2RPQs) and* conjunctive 2RPQs *(C2RPQs) are defined analogously to N2RPQs and CN2RPQs but allowing only plain REs in atoms.*

Given an interpretation $\mathcal{I}$, the semantics of an NRE E is defined by induction on its structure:

$$(E_1 \cdot E_2)^{\mathcal{I}} = E_1^{\mathcal{I}} \circ E_2^{\mathcal{I}},$$
$$(E_1 \cup E_2)^{\mathcal{I}} = E_1^{\mathcal{I}} \cup E_2^{\mathcal{I}},$$
$$(E_1^*)^{\mathcal{I}} = (E_1^{\mathcal{I}})^*,$$
$$\langle E \rangle^{\mathcal{I}} = \{(o, o) \mid \text{there is } o' \in \Delta^{\mathcal{I}} \text{ s.t. } (o, o') \in E^{\mathcal{I}}\}.$$

A *match* for a C2NRPQ $q(\vec{x}) = \exists \vec{y}.\varphi$ in an interpretation $\mathcal{I}$ is a mapping from the terms in φ to $\Delta^{\mathcal{I}}$ such that (i) $\pi(a) = a^{\mathcal{I}}$ for every individual a of φ, (ii) $\pi(x) \in A^{\mathcal{I}}$ for every concept atom $A(x)$ of φ, and (iii) $(\pi(x), \pi(y)) \in E^{\mathcal{I}}$ for every role atom $E(x, y)$ of φ. Let $\mathsf{ans}(q, \mathcal{I}) = \{\pi(\vec{x}) \mid \pi \text{ is a match for } q \text{ in } \mathcal{I}\}$. An individual tuple $\vec{a}$ with the same arity as $\vec{x}$ is called a *certain answer* to q over a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if $(\vec{a})^{\mathcal{I}} \in \mathsf{ans}(q, \mathcal{I})$ for every model $\mathcal{I}$ of $\langle \mathcal{T}, \mathcal{A} \rangle$. We use $\mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ to denote the set of all certain answers to q over $\langle \mathcal{T}, \mathcal{A} \rangle$. In what follows, by *query answering*, we will mean the problem of deciding whether $\vec{a} \in \mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$.

Example 3.1. We consider an ABox of advisor relationships of PhD holders[3]. We assume an *advisor* relation between nodes representing academics. There are also nodes for theses, universities, research topics, and countries, related in the natural way via roles *wrote*, *subm*(itted), *topic*, and *loc*(ation). We give two queries over this ABox.

$$q_1(x, y) = (advisor \cdot \langle wrote \cdot topic \cdot Physics? \rangle)^* (x, y)$$

[3]Our examples are inspired by the *Mathematics Genealogy Project* (`http://genealogy.math.ndsu.nodak.edu/`).

Query q_1 is an N2RPQ that retrieves pairs of a person x and an academic ancestor y of x such that all people on the path from x to y (including y itself) wrote a thesis in Physics.

$$q_2(x, y, z) = advisor^-(x, z), \quad advisor^*(x, w),$$
$$advisor^- \cdot \langle wrote \cdot \langle topic \cdot DBs? \rangle \cdot subm \cdot loc \cdot \{usa\}? \rangle (y, z),$$
$$\left(advisor \cdot \langle wrote \cdot \langle topic \cdot Logic? \rangle \cdot subm \cdot loc \cdot EU? \rangle\right)^* (y, w)$$

Query q_2 is a CN2RPQ that looks for triples of individuals x, y, z such that x and y have both supervised z, who wrote a thesis on Databases and who submitted this thesis to a university in the USA. Moreover, x and y have a common ancestor w, and all people on the path from x to w, including w, must have written a thesis in Logic and must have submitted this thesis to a university in an EU country. ■

It will often be more convenient to deal with an automata-based representation of (C)N2RPQs, which we provide next.

Definition 3.2. *A* nested NFA *(n-NFA) has the form* $(\mathbf{A}, s_0, F_0)$ *where* $\mathbf{A}$ *is an indexed set* $\{\alpha_1, \ldots, \alpha_n\}$*, where each* $\alpha_l \in \mathbf{A}$ *is an automaton of the form* (S, s, δ, F)*, where* S *is a set of states,* $s \in S$ *is the initial state,* $F \subseteq S$ *is the set of final states, and*

$$\delta \subseteq S \times (\mathsf{Roles} \cup \{\langle j_1, \ldots, j_k \rangle \mid l < j_i \leq n, \text{ for } i \in \{1, \ldots, k\}\}) \times S$$

We assume that the sets of states of the automata in $\mathbf{A}$ *are pairwise disjoint, and we require that* $\{s_0\} \cup F_0$ *are states of a single automaton in* $\mathbf{A}$*. If in each transition* $(s, \langle j_1, \ldots, j_k \rangle, s')$ *of each automaton in* $\mathbf{A}$ *we have* $k = 1$*, then the n-NFA is called* reduced*.*

When convenient notationally, we will denote an *n*-NFA $(\mathbf{A}, s_0, F_0)$ by $\mathbf{A}_{s_0, F_0}$. Moreover, we will use S_i, δ_i, and F_i to refer to the states, transition relation, and final states of α_i.

Definition 3.3. *Given an interpretation* $\mathcal{I}$*, we define* $\mathbf{A}^{\mathcal{I}}_{s_0, F_0}$ *inductively as follows. Let* α_l *be the (unique) automaton in* $\mathbf{A}$ *such that* $\{s_0\} \cup F_0 \subseteq S_l$*. Then* $(o, o') \in \mathbf{A}^{\mathcal{I}}_{s_0, F_0}$ *if there is a sequence* $s_0 o_0 s_1 \cdots o_{k-1} s_k o_k$*, for* $k \geq 0$*, such that* $o_0 = o$*,* $o_k = o'$*,* $s_k \in F_0$*, and for* $i \in \{1, \ldots, k\}$ *there is a transition* $(s_{i-1}, \sigma_i, s_i) \in \delta_l$ *such that either*

- $\sigma_i \in \mathsf{Roles}$ *and* $(o_{i-1}, o_i) \in \sigma_i^{\mathcal{I}}$*, or*
- $\sigma_i = \langle j_1, \ldots, j_k \rangle$ *such that, for every* $m \in \{1, \ldots, k\}$*, there exists* $o_m \in \Delta^{\mathcal{I}}$ *with* $(o_i, o_m) \in \mathbf{A}^{\mathcal{I}}_{s', F'}$*, where* s' *and* F' *are the initial and final states of* α_{j_m} *respectively.*

Note that an n-NFA $\mathbf{A}_{s_0,F_0}$ such that there are no transitions of the form $(s, \langle j_1, \ldots, j_k \rangle, s')$ in the unique α_l with $\{s_0\} \cup F_0 \subseteq S_l$ is equivalent to a standard NFA.

For every NRE E one can construct in polynomial time an n-NFA $\mathbf{A}_{s_0,F_0}$ such that $E^{\mathcal{I}} = \mathbf{A}^{\mathcal{I}}_{s_0,F_0}$ for every interpretation $\mathcal{I}$. This is an almost immediate consequence of the correspondence between regular expressions and finite state automata. Moreover, any n-NFA can be transformed into an equivalent reduced n-NFA by introducing linearly many additional states. In the following, unless stated otherwise, we assume all n-NFAs are reduced.

4 Upper Bounds via Reductions

In this section, we derive some upper bounds on the complexity of answering (C)N2RPQs in different DLs, by means of reductions to other problems. For simplicity, we assume in the rest of this section that query atoms do not employ test roles of the form $\{a\}?$. This is without loss of generality, since each symbol $\{a\}?$ can be replaced by $A_a?$ for a fresh concept name A_a, by adding the ABox assertion $A_a(a)$.

We start by showing that answering CN2RPQs can be polynomially reduced to answering non-nested C2RPQs using TBox axioms that employ inverses, conjunction on the left, and qualified existential restrictions.

Proposition 4.1. *For each CN2RPQ q, one can compute in polynomial time an $\mathcal{ELI}$ TBox $\mathcal{T}'$ and C2RPQ q' such that $\mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathsf{ans}(q', \langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \rangle)$ for every KB $\langle \mathcal{T}, \mathcal{A} \rangle$.*

Proof. Let q be an arbitrary CN2RPQ whose role atoms are given by n-NFAs, that is, they take the form $\mathbf{A}_{s_0,F_0}(x, y)$.

For each atom $\mathbf{A}_{s_0,F_0}(x, y)$ in q and each $\alpha_i \in \mathbf{A}$, we use a fresh concept name A_s for each state $s \in S_i$, and define a TBox $\mathcal{T}_{\alpha_i}$ that contains:

- $\top \sqsubseteq A_f$ for each $f \in F_i$,
- $\exists r.A_{s'} \sqsubseteq A_s$ for each $(s, r, s') \in \delta_i$ with $r \in \mathsf{N}_\mathsf{R}^\pm$,
- $A_{s'} \sqcap A \sqsubseteq A_s$ for each $(s, A?, s') \in \delta_i$ with $A \in \mathsf{N}_\mathsf{C}$, and
- $A_{s'} \sqcap A_{s_j} \sqsubseteq A_s$ for each $(s, \langle j \rangle, s') \in \delta_i$, with s_j the initial state of α_j.

We denote by $\mathcal{T}_\mathbf{A}$ the union of all $\mathcal{T}_{\alpha_i}$ with $\alpha_i \in \mathbf{A}$, and define $\mathcal{T}'$ as the union of $\mathcal{T}_\mathbf{A}$ for all atoms $\mathbf{A}_{s_0,F_0}(x, y) \in q$. To obtain the query q' we replace each atom $\mathbf{A}_{s_0,F_0}(x, y)$ by the atom $\alpha'_i(x, y)$, where α_i is the unique automaton in $\mathbf{A}$ with $\{s_0\} \cup F_0 \subseteq S_i$, and α'_i is obtained from α_i by replacing each transition of the form $(s, \langle j \rangle, s') \in \delta_i$ with $(s, A_{s_j}?, s')$, for s_j the initial state of α_j. Note that each α'_i is a standard NFA. We show in the appendix that $\mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathsf{ans}(q', \langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \rangle)$, for every KB $\langle \mathcal{T}, \mathcal{A} \rangle$. □

It follows that in every DL that contains $\mathcal{ELI}$, answering CN2RPQs is no harder than answering plain C2RPQs. From existing upper bounds for C2RPQs (Calvanese, Eiter, and Ortiz 2009; Ortiz, Rudolph, and Simkus 2011), we obtain:

Corollary 4.2. *Answering CN2RPQs is:*

- *in* 2EXP *in combined complexity for all DLs contained in $\mathcal{SHIQ}$, $\mathcal{SHOI}$, $\mathcal{ZIQ}$, or $\mathcal{ZOI}$.*
- *in* EXP *in combined complexity and* P *in data complexity for all DLs contained in Horn-$\mathcal{SHOIQ}$.*

We point out that the 2EXP upper bound for expressive DLs can also be inferred, without using the reduction above, from the existing results for answering C2RPQs in $\mathcal{ZIQ}$ and $\mathcal{ZOI}$ (Calvanese, Eiter, and Ortiz 2009).[4] Indeed, these DLs support regular role expressions as concept constructors, and a nested expression $\langle E \rangle$ in a query can be replaced by a concept $\exists E.\top$ (or by a fresh concept name A_E if the axiom $\exists E.\top \sqsubseteq A_E$ is added to the TBox). Hence, in $\mathcal{ZIQ}$ and $\mathcal{ZOI}$, nested expressions provide no additional expressiveness and CN2RPQs and C2RPQs coincide.

The construction used in Proposition 4.1 also allows us to reduce the evaluation of a N2RPQ to standard reasoning in any DL that contains $\mathcal{ELI}$.

Proposition 4.3. *For every N2RPQ q and every pair of individuals a, b, one can compute in polynomial time an $\mathcal{ELI}$ TBox $\mathcal{T}'$, and a pair of assertions $A_b(b)$ and $A_s(a)$ such that $(a, b) \in \mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ iff $\langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \{A_b(b)\} \rangle \models A_s(a)$, for every DL KB $\langle \mathcal{T}, \mathcal{A} \rangle$.*

From this and existing upper bounds for instance checking in DLs, we easily obtain:

Corollary 4.4. *Answering N2RPQs is in* EXP *in combined complexity for every DL that contains $\mathcal{ELI}$ and is contained in $\mathcal{SHIQ}$, $\mathcal{SHOI}$, $\mathcal{ZIQ}$, or $\mathcal{ZOI}$.*

5 Lower Bounds

The upper bounds we have stated in Section 4 are quite general, and in most cases worst-case optimal.

The 2EXP upper bound stated in the first item of Corollary 4.2 is optimal already for C2RPQs and $\mathcal{ALC}$. Indeed, the 2EXP hardness proof for conjunctive queries in $\mathcal{SH}$ by Eiter et al. (2009) can be adapted to use an $\mathcal{ALC}$ TBox and a C2RPQ. Also the EXP bounds in Corollaries 4.2 and 4.4 are optimal for all DLs that contain $\mathcal{ELI}$, because standard reasoning tasks like satisfiability checking are already EXP-hard in this logic (Baader, Brandt, and Lutz 2008). For the same reasons, the P bound for data complexity in Corollary 4.2 is tight for $\mathcal{EL}$ and its extensions (Calvanese et al. 2006).

However, for the lightweight DLs *DL-Lite*$_\mathcal{R}$ and $\mathcal{EL}$, the best combined complexity lower bounds we have are NL (resp., P) for N2RPQs and PSPACE for CN2RPQs, inherited from the lower bounds for (C)NRPQs (Bienvenu, Ortiz, and Simkus 2013). This leaves a significant gap with respect to the EXP upper bounds in Corollaries 4.2 and 4.4.

We show next that these upper bounds are tight. This is the one of the core technical results of this paper, and probably the most surprising one: already evaluating one N2RPQ in the presence of a *DL-Lite* or $\mathcal{EL}$ TBox is EXP-hard.

Theorem 5.1. *In DL-Lite and $\mathcal{EL}$, N2RPQ answering is* EXP*-hard in combined complexity.*

Proof. We provide a reduction from the word problem for *Alternating Turing Machines (ATMs)* with polynomially

[4] For queries that do not contain inverse roles, that is, (1-way) CRPQs, the same applies to $\mathcal{ZOQ}$ and its sublogics.

bounded space, which is known to be EXP-hard (Chandra, Kozen, and Stockmeyer 1981). An ATM is given as a tuple $M = (\Sigma, S_\exists, S_\forall, \delta, s_{init}, s_{acc}, s_{rej})$, where Σ is an *alphabet*, $S_\exists$ is a set of *existential states*, $S_\forall$ is a set of *universal states*, $\delta \subseteq (S_\exists \cup S_\forall) \times \Sigma \cup \{\mathfrak{b}\} \times (S_\exists \cup S_\forall) \times \Sigma \cup \{\mathfrak{b}\} \times \{-1, 0, +1\}$ is a *transition relation*, $\mathfrak{b}$ is the blank symbol, and $s_{init}, s_{acc}, s_{rej} \in S_\exists$ are the *initial state*, the *acceptance state* and the *rejection state*, respectively.

Consider a word $w \in \Sigma^*$. We can w.l.o.g. assume that $\Sigma = \{0, 1\}$, that M uses only $|w|$ tape cells and that $|w| \geq 1$. Let $m = |w|$, and, for each $1 \leq i \leq m$, let $w(i)$ denote the ith symbol of w. Let $S = S_\exists \cup S_\forall$. We make the following further assumptions:

(i) The initial state is not a final state: $s_{init} \notin \{s_{acc}, s_{rej}\}$.
(ii) Before entering a state s_{acc} or s_{rej}, M writes $\mathfrak{b}$ in all m tape cells.
(iii) There exist functions $\delta_1, \delta_2 : S \times \Sigma \cup \{\mathfrak{b}\} \to S \times \Sigma \cup \{\mathfrak{b}\} \times \{-1, 0, +1\}$ such that $\{\delta_1(s, \sigma), \delta_2(s, \sigma)\} = \{(s', \sigma', d) \mid (s, \sigma, s', \sigma', d) \in \delta\}$ for every $s \in S \setminus \{s_{acc}, s_{rej}\}$ and $\sigma \in \Sigma \cup \{\mathfrak{b}\}$. In other words, non-final states of M give rise to exactly two successor configurations described by the functions δ_1, δ_2.

Note that the machine M can be modified in polynomial time to ensure (i-iii), while preserving the acceptance of w.

We next show how to construct in polynomial time a *DL-Lite* KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a query q such that M accepts w iff $a \in \mathsf{ans}(q, \mathcal{K})$ (we return to $\mathcal{EL}$ later). The high-level idea underlying the reduction is to use a KB to enforce a tree that contains all possible computations of M on w. The query q selects a computation in this tree and verifies that it corresponds to a proper, error-free, accepting run.

Generating the tree of transitions. First we construct $\mathcal{K}$, which enforces a tree whose edges correspond to the possible transitions of M. More precisely, each edge encodes a transition together with the resulting position of the read/write head of M, and indicates whether the transition is given by δ_1 or δ_2. This is implemented using role names $r_{p,t,i}$, where $p \in \{1, 2\}$, $t \in \delta$, and $0 \leq i \leq m + 1$. To mark the nodes that correspond to the initial (resp., a final) configuration of M, we employ the concept name A_{init} (resp., A_{final}), and we use $A_\exists$ and $A_\forall$ to store the transition type.

We let $\mathcal{A} = \{A_{init}(a), A_\exists(a)\}$, and then we initiate the construction of the tree by including in $\mathcal{T}$ the axiom

$$A_{init} \sqsubseteq \exists r_{p,(s_{init},\sigma,s',\sigma',d),1+d}. \tag{1}$$

for each $\sigma \in \Sigma \cup \{\mathfrak{b}\}$ and $p \in \{1, 2\}$ such that $\delta_p(s_{init}, \sigma) = (s', \sigma', d)$. To generate further transitions, $\mathcal{T}$ contains

$$\exists r^-_{p,(s,\sigma,s',\sigma',d),i} \sqsubseteq \exists r_{p',(s',\sigma^*,s'',\sigma'',d'),i+d'} \tag{2}$$

for each $(s, \sigma, s', \sigma', d) \in \delta$, $1 \leq i \leq m$, $\sigma^* \in \Sigma \cup \{\mathfrak{b}\}$ and $p, p' \in \{1, 2\}$ such that $\delta_{p'}(s', \sigma^*) = (s'', \sigma'', d')$. Note that a transition $t' = (s', \sigma^*, s'', \sigma'', d') \in \delta$ can follow $t = (s, \sigma, s', \sigma', d) \in \delta$ only if σ^* is the symbol written on tape cell i, for i the position of the read/write head after executing t. This is not guaranteed by (2). Instead, we "overestimate" the possible successive transitions, and use the query q to select paths that correspond to a proper computation.

We complete the definition of $\mathcal{T}$ by adding inclusions to label the nodes according to the type of states resulting from transitions. For each $1 \leq i \leq m$, $p \in \{1, 2\}$ and transition $(s, \sigma, s', \sigma', d) \in \delta$, we have the axiom

$$\exists r^-_{p,(s,\sigma,s',\sigma',d),i} \sqsubseteq A_Q, \text{ where}$$

- $A_Q = A_{final}$ if $s' \in \{s_{acc}, s_{rej}\}$,
- $A_Q = A_\exists$, if $s' \in S_\exists \setminus \{s_{acc}, s_{rej}\}$, and
- $A_Q = A_\forall$, if $s' \in S_\forall \setminus \{s_{acc}, s_{rej}\}$.

We turn to the construction of the query q, for which we employ the *n*-NFA representation. We construct an *n*-NFA $\alpha_q = (\mathbf{A}, s, F)$ where $\mathbf{A}$ has $m+1$ automata $\{\alpha_0, \ldots, \alpha_m\}$. Intuitively, the automaton α_0 will be responsible for traversing the tree representing candidate computation paths. At nodes corresponding to the end of a computation path, α_0 launches $\alpha_1, \ldots, \alpha_m$ which "travel" back to the root of the tree and test for the absence of errors along the way. We start by defining the tests $\alpha_1, \ldots, \alpha_m$. Afterwards we define α_0, which selects a set of paths that correspond to a full computation, and launches these tests at the end of each path.

Testing the correctness of a computation path. For each $1 \leq l \leq m$, the automaton $\alpha_l = (S_l, s_l, \delta_l, F_l)$ is built as follows. We let $S_l = \{\sigma_l \mid \sigma \in \Sigma\} \cup \{\mathfrak{b}_l\} \cup \{s'_l\}$. That is, S_l contains a copy of $\Sigma \cup \{\mathfrak{b}\}$ plus the additional state s'_l. We define the initial state as $s_l = \mathfrak{b}_l$ and let $F_l = \{s'_l\}$. Finally, the transition relation δ_l contains the following tuples:

(T1) $(\sigma_l, r^-_{p,(s,\sigma,s',\sigma',d),i}, \sigma_l)$ for all $1 \leq i \leq m$, $p \in \{1, 2\}$, all transitions $(s, \sigma, s', \sigma', d) \in \delta$, and each $\sigma_l \in S_l \setminus \{s'_l\}$ with $l \neq i - d$;
(T2) $(\sigma'_l, r^-_{p,(s,\sigma,s',\sigma',d),i}, \sigma_l)$ for all $1 \leq i \leq m$, $s \in S$ and $p \in \{1, 2\}$ with $\delta_p(s, \sigma) = (s', \sigma', d)$ and $l = i - d$;
(T3) $(\sigma_l, A_{init}?, s'_l)$ for $\sigma = w(l)$.

The working of α_l can be explained as follows. Each state $\sigma_l \in S_l \setminus \{s'_l\}$ corresponds to one of the symbols that may be written in position l of the tape during a run of M. When α_l is launched at some node in a computation tree induced by $\mathcal{K}$, it attempts to travel up to the root node, and the only reason it may fail is when a wrong symbol is written in position l at some point in the computation path. Recall that in each final configuration of M, all symbols are set to the blank symbol, and thus the initial state of α_l is $\mathfrak{b}_l$.

Consider a word $w' \in \mathsf{Roles}^*$ of the form

$$r^-_{p_k,t_k,i_k} \cdots r^-_{p_1,t_1,i_1} \cdot A_{init}? \tag{3}$$

that describes a path from some node in the tree induced by $\mathcal{K}$ up to the root node a. We claim that w' is accepted by every α_l ($1 \leq l \leq m$) just in the case that $t_1, \ldots, t_k$ is a correct sequence of transitions. To see why, first suppose that every α_l accepts w', and let $(pos_0, st_0, tape_0)$ be the tuple with $pos_0 = 1$, $st_0 = s_{init}$ and $tape_0$ contains for each $1 \leq l \leq m$, the symbol σ_l corresponding to the state of α_l when reading A_{init}. Clearly, due to (T3), the tuple $(pos_0, st_0, tape_0)$ describes the initial configuration of M on input w. For $1 \leq j \leq k$, if $t_j = (s, \sigma, s', \sigma', d)$, then we define $(pos_j, st_j, tape_j)$ as follows: $pos_j = i_j$, $st_j = s'$, and $tape_j$ contains for each $1 \leq i \leq m$, the state of α_i when reading $r^-_{p_j,t_j,i_j}$. A simple inductive argument

shows that for every $1 \leq j \leq k$, the tuple $(pos_j, st_j, tape_j)$ describes the configuration of M after applying the transitions $t_1, \ldots, t_j$ from the initial configuration. Indeed, let us assume that $(pos_{j-1}, st_{j-1}, tape_{j-1})$ correctly describes the configuration after executing $t_1, \ldots, t_{j-1}$ and $t_j = (s, \sigma, s', \sigma', d)$. After executing t_j, the read/write head is in position $pos_{j-1} + d$ and the state is s'. Since the only way to enforce an $r^-_{p_j,t_j,i_j}$-edge is via axioms (1) and (2), we must have $pos_j = pos_{j-1} + d$ and $st_j = s'$. It remains to show that $tape_j$ describes the tape contents after executing t_j. Consider some position $1 \leq l \leq m$. There are two cases:

1. $l \neq i_j - d$. In this case, we know that the symbol in position l is not modified by executing t_j. We have to show that $\sigma_l \in tape_{j-1}$ implies $\sigma_l \in tape_j$. This follows from the construction of α_l. In particular, when reading $r_{p_j,t_j,i_j}{}^-$, it must employ a transition from (T1).
2. $l = i_j - d$. In this case, after executing t_j, we must have σ' in position l. We have to show that $\sigma_l \in tape_{j-1}$ implies $\sigma'_l \in tape_j$. This again follows from the construction of α_l. In particular, when reading $r_{p_j,t_j,i_j}{}^-$, there is only one possible transition available in (T2), namely $(\sigma'_l, r_{p_j,t_j,i_j}{}^-, \sigma_l)$.

Conversely, it is easy to see that any word of the form (3) that appears in the tree induced by $\mathcal{K}$ and represents a correct computation path will be accepted by all of the α_l.

Selecting a proper computation. It remains to define α_0, which selects a subtree corresponding to a full candidate computation of M, and then launches the tests defined above at the end of each path. We let $\alpha_0 = (S_0, s_0, \delta_0, F_0)$, where $S_0 = \{s_\downarrow, c_L, c_R, s_\uparrow, s_\updownarrow, s_{test}, s_f\}$, $s_0 = s_\downarrow$, $F_0 = \{s_f\}$, and δ_0 is defined next.

The automaton operates in two main modes: moving down the tree away from the root and moving back up towards the root. Depending on the type of the state of M, in state $s_\downarrow$ the automaton either selects a child node to process next, or chooses to launch the test automata. If the tests are successful, it switches to moving up. To this end, δ_0 has the following transitions:

$$(s_\downarrow, A_\exists?, c_L), (s_\downarrow, A_\exists?, c_R), (s_\downarrow, A_\forall?, c_L),$$
$$(s_\downarrow, A_{final}?, s_{test}), \text{ and } (s_{test}, \langle 1, \ldots, m\rangle, s_\uparrow).$$

The transitions that implement a step down or up are:

- $(c_L, r_{1,t,i}, s_\downarrow)$ for every $1 \leq i \leq m$ and $t \in \delta$,
- $(c_R, r_{2,t,i}, s_\downarrow)$ for every $1 \leq i \leq m$ and $t \in \delta$,
- $(s_\uparrow, r^-_{1,t,i}, s_\updownarrow)$ for every $1 \leq i \leq m$ and $t \in \delta$, and
- $(s_\uparrow, r^-_{2,t,i}, s_\uparrow)$ for every $1 \leq i \leq m$ and $t \in \delta$.

After making a step up from the state $s_\uparrow$ via an $r^-_{1,t,i}$-edge, the automaton enters the state $s_\updownarrow$. Depending on the encountered state of M, the automaton decides either to verify the existence of a computation tree for the alternative transition, to keep moving up, or to accept the word. This is implemented using the following transitions of δ_0:

$$(s_\updownarrow, ?A_\forall, c_R), (s_\updownarrow, ?A_\exists, s_\uparrow), \text{ and } (s_\updownarrow, ?A_{init}, s_f).$$

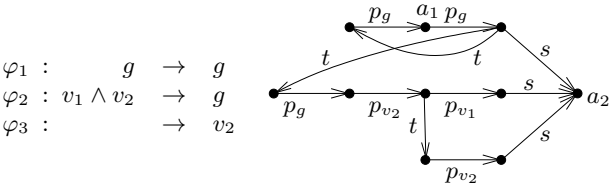

Figure 1: Example ABox in the proof of Theorem 5.2

To conclude the definition of $\alpha_q = (\mathbf{A}, s, F)$, set $s = s_\downarrow$ and $F = \{s_f\}$. Note that α_q has a constant number of states, so it can be converted into an equivalent NRE E_q in polynomial time. The desired query is $q(x, y) = E_q(x, y)$.

The above *DL-Lite* TBox $\mathcal{T}$ can be easily rephrased in $\mathcal{EL}$. Indeed, we simply take a fresh concept name $A_{p,t,i}$ for each role $r_{p,t,i}$, and replace every axiom $C \sqsubseteq \exists r_{p,t,i}$ by $C \sqsubseteq \exists r_{p,t,i}.A_{p,t,i}$ and every axiom $\exists r^-_{p,t,i} \sqsubseteq C$ by $A_{p,t,i} \sqsubseteq C$. $\square$

The above lower bound for answering N2RPQs hinges on the support for existential concepts in the right-hand-side of inclusions. If they are disallowed, then one can find a polynomial time algorithm (Pérez, Arenas, and Gutierrez 2010). However, it was open until now whether the polynomial-time upper bound is optimal. We next prove P-hardness of the problem, already for plain graph databases.

Theorem 5.2. *Given as input an N2RPQ q, a finite interpretation $\mathcal{I}$ and a pair $(o, o') \in \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, it is* P*-hard to check whether $(o, o') \in \mathsf{ans}(q, \mathcal{I})$.*

Proof. To simplify the presentation, we prove the lower bound for a slight reformulation of the problem. In particular, we show P-hardness of deciding $\vec{c} \in \mathsf{ans}(q, \langle \emptyset, \mathcal{A}\rangle)$, where q is an N2RPQ and $\mathcal{A}$ is an ABox with assertions only of the form $A(a)$ or $r(a, b)$, where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$.

We provide a logspace reduction from the classical P-complete problem of checking entailment in propositional definite Horn theories. Assume a set $T = \{\varphi_1, \ldots, \varphi_n\}$ of definite clauses over a set of propositional variables V, where each φ_i is represented as a rule $v_1 \wedge \ldots \wedge v_m \rightarrow v_{m+1}$.

Given a variable $g \in V$, we define an ABox $\mathcal{A}$, an N2RPQ q, and tuple (a_1, a_2) such that $T \models g$ iff $(a_1, a_2) \in \mathsf{ans}(q, \langle \emptyset, \mathcal{A}\rangle)$. We may assume w.l.o.g. that $\varphi_1 = g \rightarrow g$. We define the desired ABox as $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, using the role names s, t, and p_v, where $v \in V$. The ABox $\mathcal{A}_1$ simply encodes T and contains for every $\varphi_i = v_1 \wedge \ldots \wedge v_m \rightarrow v_{m+1}$, the following assertions:

$$p_{v_{m+1}}(e^i_{m+1}, e^i_m), \ldots, p_{v_1}(e^i_1, e^i_0), s(e^i_0, f).$$

The ABox $\mathcal{A}_2$ links variables in rule bodies with their occurrences in rule heads. For every pair of rules $\varphi_i = v_1 \wedge \ldots \wedge v_m \rightarrow v_{m+1}$ and $\varphi_j = w_1 \wedge \ldots \wedge w_n \rightarrow w_{n+1}$, and each $1 \leq l \leq m$ with $v_l = w_{n+1}$, it contains the assertion $t(e^i_{l-1}, e^j_{n+1})$. See Figure 1 for an example.

The existence of a proof tree for g, which can be limited to depth $|V|$, is expressed using the query $q(x, y) = E_{|V|}(x, y)$, with $E_1, E_2, \ldots, E_{|V|}$ defined inductively:

$$E_1 = \bigcup_{v \in V} (p_v \cdot t \cdot p_v) \cdot s$$

$$E_i = \bigcup_{v \in V} \big(p_v \cdot t \cdot p_v\big) \cdot \big(\langle E_{i-1} \rangle \cdot \bigcup_{v \in V} p_v\big)^* \cdot s \qquad (i > 1)$$

Finally, we let $a_1 = e_1^1$ and $a_2 = f$. □

6 Concrete Approach for Horn DLs

Our complexity results so far leave a gap for the data complexity of the *DL-Lite* family: we inherit NL-hardness from plain RPQs, but we only have the P upper bound stemming from Proposition 4.1. In this section, we close this gap by providing an NL upper bound.

This section has an additional goal. We recall that the upper bounds in Corollaries 4.2 and 4.4 rely on reductions to answering (C)2RPQs in extensions of $\mathcal{ELI}$, like Horn-$\mathcal{SHOIQ}$, $\mathcal{ZIQ}$, and $\mathcal{ZOI}$. Unfortunately, known algorithms for C2RPQ answering in these logics use automata-theoretic techniques that are best-case exponential and not considered suitable for implementation. Hence, we want to provide a direct algorithm that may serve as a basis for practicable techniques. To this end, we take an existing algorithm for answering C2RPQs in $\mathcal{ELH}$ and *DL-Lite*$_\mathcal{R}$ due to Bienvenu et al. (2013) and show how it can be extended to handle CN2RPQs and $\mathcal{ELHI}_\bot$ KBs.

For presenting the algorithm in this section, it will be useful to first recall the canonical model property of $\mathcal{ELHI}_\bot$.

Canonical Models

We say that an $\mathcal{ELHI}_\bot$ TBox $\mathcal{T}$ is in *normal form* if all of its concept inclusions are of one of the following forms:

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

with $A, B, B_1, B_2 \in \mathsf{N_C}$ and $r \in \mathsf{N_R^\pm}$.

By introducing fresh concept names to stand for complex concepts, every TBox $\mathcal{T}$ can be transformed in polynomial time into a TBox $\mathcal{T}'$ in normal form that is a model-conservative extension of $\mathcal{T}$. Hence, in what follows, we assume that $\mathcal{ELHI}_\bot$ TBoxes are in normal form.

The domain of the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ of a consistent KB $\langle \mathcal{T}, \mathcal{A} \rangle$ consists of all sequences $ar_1C_1 \dots r_nC_n$ ($n \geq 0$) such that:

- $a \in \mathsf{Ind}(\mathcal{A})$ and $r_i \in \mathsf{N_R^\pm}$ for each $1 \leq i \leq n$;
- each C_i is a finite conjunction of concept names;
- if $n \geq 1$, then $\mathcal{T}, \mathcal{A} \models (\exists r_1.C_1)(a)$;
- for $1 \leq i < n$, $\mathcal{T} \models C_i \sqsubseteq \exists r_{i+1}.C_{i+1}$.

For an $o \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A})$, we use $\mathsf{tail}(o)$ to denote its final concept. The interpretation $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is then defined as follows:

$$\begin{aligned}
a^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= a \text{ for all } a \in \mathsf{Ind}(\mathcal{A}) \\
A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T}, \mathcal{A} \models A(a)\} \\
&\quad \cup \{o \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T} \models \mathsf{tail}(o) \sqsubseteq A\} \\
p^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{(a,b) \mid p(a,b) \in \mathcal{A}\} \cup \\
&\quad \{(o_1,o_2) \mid o_2 = o_1 r\, C \text{ and } \mathcal{T} \models r \sqsubseteq p\} \cup \\
&\quad \{(o_2,o_1) \mid o_2 = o_1 r\, C \text{ and } \mathcal{T} \models r \sqsubseteq p^-\}
\end{aligned}$$

Observe that $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is composed of a core part containing the individuals from $\mathcal{A}$ and an *anonymous part* consisting of (possibly infinite) trees rooted at the ABox individuals. We use $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_o$ to denote the restriction of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ to domain elements having o as a prefix.

It is well-known that the canonical model of a consistent $\mathcal{ELHI}_\bot$ KB $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ can be homomorphically embedded into any model of $\langle \mathcal{T}, \mathcal{A} \rangle$. Since CN2RPQs are preserved under homomorphisms, we have:

Lemma 6.1. *For every consistent $\mathcal{ELHI}_\bot$ KB $\langle \mathcal{T}, \mathcal{A} \rangle$, CN2RPQ q, and tuple $\vec{a}$ of individuals: $\vec{a} \in \mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ if and only if $\vec{a} \in \mathsf{ans}(q, \mathcal{I}_{\mathcal{T},\mathcal{A}})$.*

Computing Jump and Final Transitions

A crucial component of our algorithm is to compute relevant partial paths in a subtree $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_o$ rooted at an object o in the anonymous part of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. Importantly, we also need to remember which parts of the nested automata that have been partially navigated below o still need to be continued. This will allow us to 'forget' the tree below o.

In what follows, it will be convenient use *runs* to talk about the semantics of *n*-NFAs.

Definition 6.1. *Let $\mathcal{I}$ be an interpretation, and let $(\mathbf{A}, s_0, F_0)$ be an n-NFA. Then a* partial run for $\mathbf{A}$ on $\mathcal{I}$ *is a finite node-labelled tree (T, ℓ) such that every node is labelled with an element from $\Delta^\mathcal{I} \times (\bigcup_i S_i)$ and for each non-leaf node v having label $\ell(v) = (o, s)$ with $s \in S_i$, one of the following holds:*

- *v has a unique child v' with $\ell(v') = (o', s')$, and there exists $(s, \sigma, s') \in \delta_i$ such that $\sigma \in \mathsf{Roles}$ and $(o, o') \in \sigma^\mathcal{I}$;*
- *v has exactly two children v' and v'' with $\ell(v') = (o, s')$ and $\ell(v'') = (o, s'')$, with s'' the initial state of α_j, and there exists a transition $(s, \langle j \rangle, s') \in \delta_i$.*

*If T has root labelled (o_1, s_1) and a leaf node labelled (o_2, s_2) with s_1, s_2 states of the same α_i, then (T, ℓ) is called an (o_1, s_1, o_2, s_2)-*run*, and it is* full *if every leaf label $(o', s') \neq (o_2, s_2)$ is such that $s' \in F_k$ for some k.*

Full runs provide an alternative characterization of the semantics of *n*-NFAs in Definition 3.3.

Fact 6.2. *For every interpretation $\mathcal{I}$, $(o_1, o_2) \in (\mathbf{A}_{s_1,\{s_2\}})^\mathcal{I}$ if and only if there is a full (o_1, s_1, o_2, s_2)-run for $\mathbf{A}$ in $\mathcal{I}$.*

We use partial runs to characterize when an *n*-NFA $\mathbf{A}$ can be partially navigated inside a tree $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_o$ whose root satisfies some conjunction of concepts C. Intuitively, $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ stores pairs s_1, s_2 of states of some $\alpha \in \mathbf{A}$ such that a path from s_1 to s_2 exists, while $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ stores states s_1 for which a path to some final state exists, no matter where the final state is reached. Both $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ and $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ store a set Γ of states s of other automata nested in α, for which a path from s to a final state remains to be found.

Definition 6.2. *Let $\mathcal{T}$ be an $\mathcal{ELHI}_\bot$ TBox in normal form and $(\mathbf{A}, s_0, F_0)$ an n-NFA. The set $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ consists of tuples (C, s_1, s_2, Γ) where C is either $\top$ or a conjunction of concept names from $\mathcal{T}$, s_1 and s_2 are states from $\alpha_i \in \mathbf{A}$, and $\Gamma \subseteq \bigcup_{j>i} S_j$. A tuple (C, s_1, s_2, Γ) belongs to $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ if there exists a partial run (T, ℓ) of $\mathbf{A}$ in the canonical model of $\langle \mathcal{T}, \{C(a)\} \rangle$ that satisfies the following conditions:*

- *the root of T is labelled (a, s_1);*
- *there is a leaf node v with $\ell(v) = (a, s_2)$;*
- *for every leaf node v with $\ell(v) = (o, s) \neq (a, s_2)$, either $s \in F_j$ for some $j > i$, or $o = a$ and $s \in \Gamma$.*

The set $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ contains all tuples (C, s_1, F, Γ) there is a partial run (T, ℓ) of $\mathbf{A}$ in the canonical model of $\langle \mathcal{T}, \{C(a)\}\rangle$ that satisfies the following conditions:

- *the root of T is labelled (a, s_1);*
- *there is a leaf node v with $\ell(v) = (o, s_f)$ and $s_f \in F$;*
- *for every leaf node v with $\ell(v) = (o, s)$, either s is a final state in some α_k, or $o = a$ and $s \in \Gamma$.*

Proposition 6.3. *It can be decided in exponential time if a tuple belongs to $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ or $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$.*

Proof idea. We first show how to use TBox reasoning to decide whether $(C, s_1, s_2, \Gamma) \in \mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$. For every $\alpha_j \in \mathbf{A}$, we introduce a fresh concept name A_s for each state $s \in S_j$. Intuitively, A_s expresses that there is an outgoing path that starts in s and reaches a final state. If $\{s_1, s_2\} \subseteq S_i$, then we add the following inclusions to $\mathcal{T}$:

- $\top \sqsubseteq A_s$, for every $s \in F_j$ with $j > i$;
- $\exists r.A_{s'} \sqsubseteq A_s$, whenever $(s, r, s') \in \delta_i$ with $r \in \mathsf{N}_\mathsf{R}^\pm$;
- $A_{s'} \sqcap B \sqsubseteq A_s$, whenever $(s, B?, s') \in \delta_i$;
- $A_{s'} \sqcap A_{s''} \sqsubseteq A_s$, whenever $(s, \langle j \rangle, s') \in \delta_i$ and s'' is the initial state of α_j.

Let $\mathcal{T}'$ be the resulting TBox. In the long version, we show that $(C, s_1, s_2, \Gamma) \in \mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ iff

$$\mathcal{T}' \models (C \sqcap A_{s_2} \sqcap \prod_{s \in \Gamma} A_s) \sqsubseteq A_{s_1}.$$

To decide if $(C, s_1, F, \Gamma) \in \mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$, we must also include in $\mathcal{T}'$ the following inclusions:

- $\top \sqsubseteq A_s$, for every $s \in F$.

We then show that $(C, s_1, F, \Gamma) \in \mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ iff

$$\mathcal{T}' \models (C \sqcap \prod_{s \in \Gamma} A_s) \sqsubseteq A_{s_1}.$$

To conclude the proof, we simply note that both problems can be decided in single-exponential time, as TBox reasoning in $\mathcal{ELHI}_\bot$ is known to be EXP-complete. □

Query Rewriting

The core idea of our query answering algorithm is to rewrite a given CN2RPQ q into a set of queries Q such that the answers to q and the union of the answers for all $q' \in Q$ coincide. However, for evaluating each $q' \in Q$, we only need to consider mappings from the variables to the individuals in the core of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. Roughly, a rewriting step makes some assumptions about the query variables that are mapped deepest into the anonymous part and, using the structure of the canonical model, generates a query whose variables are matched one level closer to the core. Note that, even when we assume that no variables are mapped below some element o in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, the satisfaction of the regular paths may require to go below o and back up in different ways. This is handled using jump and final transitions. The query rewriting algorithm is an adaptation of the algorithm for C2RPQs in (Bienvenu, Ortiz, and Simkus 2013), to which the reader may refer for more detailed explanations and examples.

The query rewriting algorithm is presented in Figure 2. In the algorithm, we use atoms of the form $\langle A_{s,F} \rangle(x)$, which are semantically equivalent to $A_{s,F}(x, z)$ for a variable z not occurring anywhere in the query. This alternative notation will spare us additional variables and make the complexity arguments simpler. To slightly simplify the notation, we may write $\mathbf{A}_{s,s'}$ instead of $\mathbf{A}_{s,\{s'\}}$.

The following proposition states the correctness of the rewriting procedure. Its proof follows the ideas outlined above and can be found in the appendix of the long version. Slightly abusing notation, we will also use $\mathsf{Rewrite}(q, \mathcal{T})$ to denote the set all of queries that can be obtained by an execution of the rewriting algorithm on q and $\mathcal{T}$.

Proposition 6.4. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHI}_\bot$ KB and $q(\vec{x})$ a C2NRPQ. Then $\vec{a} \in \mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ iff there exists $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ and a match π for q' in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi(\vec{x}) = \vec{a}$ and $\pi(y) \in \mathsf{Ind}(\mathcal{A})$ for every variable y in q'.*

We note that the query rewriting does not introduce fresh terms. Moreover, it employs an at most quadratic number of linearly sized *n*-NFAs, obtained from the *n*-NFAs of the input query. Thus, the size of each $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ is polynomial in the size of q and $\mathcal{T}$. Given that all the employed checks in Figure 2 can be done in exponential time (see Proposition 6.3), we obtain the following.

Proposition 6.5. *The set $\mathsf{Rewrite}(q, \mathcal{T})$ can be computed in exponential time in the size of q and $\mathcal{T}$.*

Query Evaluation

In Figure 3, we present an algorithm EvalAtom for evaluating N2RPQs. The idea is similar to the standard nondeterministic algorithm for deciding reachability: we guess a sequence $(c_0, s_0)(c_1, s_1) \cdots (c_m, s_m)$ of individual-state pairs, keeping only two successive elements in memory at any time. Every element (c_{i+1}, s_{i+1}) must be reached from the preceding element (c_i, s_i) by a single normal, jump, or final transition. Moreover, in order to use a jump or final transition, we must ensure that its associated conditions are satisfied. To decide if the current individual belongs to C, we can employ standard reasoning algorithms, but to determine whether an outgoing path exists for one of the states in Γ, we must make a recursive call to EvalAtom. Importantly, these recursive calls involve "lower" automata, and so the depth of recursion is bounded by the number of automata in the N2RPQ (and so is independent of $\mathcal{A}$). It follows that the whole procedure can be implemented in non-deterministic logarithmic space in $|\mathcal{A}|$, if we discount the concept and role membership tests. By exploiting known complexity results for instance checking in *DL-Lite*$_\mathcal{R}$ and $\mathcal{ELHI}_\bot$, we obtain:

Proposition 6.6. EvalAtom *is a sound and complete procedure for N2RPQ evaluation over satisfiable $\mathcal{ELHI}_\bot$ KBs. It can be implemented so as to run in non-deterministic logarithmic space (resp., polynomial time) in the size of the ABox for DL-Lite*$_\mathcal{R}$ *(resp., $\mathcal{ELHI}_\bot$) KBs.*

PROCEDURE Rewrite

Input: CN2RPQ q, $\mathcal{ELHI}_\bot$ TBox $\mathcal{T}$ in normal form

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 a conjunction C of concept names from $\mathcal{T}$ 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 $at \in q$ of the form $\langle \mathbf{A}_{s_0,F} \rangle(t)$ or $\mathbf{A}_{s_0,F}(t,t')$ with $y \in \{t,t'\}$:
   (a) let $\alpha_i \in \mathbf{A}$ be the automaton containing s_0, F
   (b) choose a sequence $s_1, \dots, s_{n-1}$ of distinct states from S_i and some $s_n \in F$
   (c) replace at by the atoms $\mathbf{A}_{s_0,s_1}(t,y)$, $\mathbf{A}_{s_1,s_2}(y,y)$, $\dots$, $\mathbf{A}_{s_{n-2},s_{n-1}}(y,y)$, and
   - $\mathbf{A}_{s_{n-1},s_n}(y,t')$ if $at = \mathbf{A}_{s_0,F}(t,t')$, or
   - $\langle \mathbf{A}_{s_{n-1},s_n} \rangle(y)$ if $at = \langle \mathbf{A}_{s_0,F} \rangle(y)$.
5. For each atom at_j of the form $\mathbf{A}_{s_j,s_{j+1}}(y,y)$ or $\langle \mathbf{A}_{s_j,s_{j+1}} \rangle(y)$ in q, either do nothing, or:
   - Choose some $(C, s_j, s_{j+1}, \Gamma) \in \mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ if $at_j = \mathbf{A}_{s_j,s_{j+1}}(y,y)$.
   - Choose some $(C, s_j, \{s_{j+1}\}, \Gamma) \in \mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ if $at_j = \langle \mathbf{A}_{s_j,s_{j+1}} \rangle(y)$.
   - Replace at_j by $\{\langle \mathbf{A}_{u,F_k} \rangle(y) \mid u \in \Gamma \cap S_k, \}$.
6. Choose a conjunction D of concept names from $\mathcal{T}$ and $r, r_1, r_2 \in \mathsf{N}_\mathsf{R}^\pm$ such that:
   (a) $\mathcal{T} \models D \sqsubseteq \exists r.C$, $\mathcal{T} \models r \sqsubseteq r_1$, and $\mathcal{T} \models r \sqsubseteq r_2$.
   (b) For each atom $\mathbf{A}_{u,U}(y,t)$ of q with $u \in S_i$, there exists $v \in S_i$ such that $(u, r_1^-, v) \in \delta_i$
   (c) For each atom $\mathbf{A}_{u,U}(t,y)$ of q with $u \in S_i$, there exists $v \in S_i$ and $v' \in U$ with $(v, r_2, v') \in \delta_i$.
   (d) For each atom $\langle \mathbf{A}_{u,U} \rangle(y)$ of q with $u \in S_i$, there exists $v \in S_i$ such that $(u, r_1^-, v) \in \delta_i$.

   For atoms $\mathbf{A}_{u,U}(y,y)$, both (b) and (c) apply.
7. Replace
   - each atom $\mathbf{A}_{u,U}(y,t)$ with $t \neq y$ by $\mathbf{A}_{v,U}(y,t)$,
   - each atom $\mathbf{A}_{u,U}(t,y)$ with $t \neq y$ by $\mathbf{A}_{u,v}(y,t)$,
   - each atom $\mathbf{A}_{u,U}(y,y)$ by atom $\mathbf{A}_{v,v'}(y,y)$, and
   - each atom $\langle \mathbf{A}_{u,U} \rangle(y)$ by atom $\langle \mathbf{A}_{v,U} \rangle(y)$

   with v, v' as in Step 6.
8. Add $A(y)$ to q for each $A \in D$ and return to Step 1.

Figure 2: Query rewriting procedure Rewrite.

We present in Figure 4 the complete procedure EvalQuery for deciding CN2RPQ entailment.

Theorem 6.7. EvalQuery *is a sound and complete procedure for deciding CN2RPQ entailment over $\mathcal{ELHI}_\bot$ KBs. In the case of DL-Lite$_\mathcal{R}$ KBs, it runs in non-deterministic logarithmic space in the size of the ABox.*

Proof idea. Soundness, completeness, and termination of

PROCEDURE EvalAtom

Input: n-NFA $(\mathbf{A}, s_0, F_0)$, $\mathcal{ELHI}_\bot$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in normal form, $(a,b) \in \mathsf{Ind}(\mathcal{A}) \times (\mathsf{Ind}(\mathcal{A}) \cup \{\mathsf{anon}\})$

1. Let i be such that $s_0 \in S_i$, and set $\mathsf{max} = |\mathcal{A}| \times |S_i| + 1$.
2. Initialize $\mathsf{current} = (a, s_0)$ and $\mathsf{count} = 0$.
3. While $\mathsf{count} < \mathsf{max}$ and $\mathsf{current} \neq (b, s_f)$ for $s_f \in F_0$
   (a) Let $\mathsf{current} = (c, s)$.
   (b) Guess a pair $(d, s') \in (\mathsf{Ind}(\mathcal{A}) \cup \{\mathsf{anon}\}) \times S_i$ such that one of the following holds:
      i. $d \in \mathsf{Ind}(\mathcal{A})$ and there exists $(s, \sigma, s') \in \delta_i$ with $\sigma \in \mathsf{Roles}$ such that $(c,d) \in \sigma^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$
      ii. $d = c$ and $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ contains a tuple (C, s, s', Γ) such that $c \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and for every $j > i$ and every $u \in \Gamma \cap S_j$,
      $$\mathsf{EvalAtom}((\mathbf{A}, u, F_j), \mathcal{K}, (c, \mathsf{anon})) = \mathtt{yes}$$
      iii. $d = \mathsf{anon}$, $s' \in F_0$, and $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ contains a tuple (C, s, F_0, Γ) such that $c \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and for every $j > i$ and every $u \in \Gamma \cap S_j$,
      $$\mathsf{EvalAtom}((\mathbf{A}, u, F_j), \mathcal{K}, (c, \mathsf{anon})) = \mathtt{yes}$$
   (c) Set $\mathsf{current} = (d, s')$ and increment count.
4. If $\mathsf{current} = (d, s_f)$ for some $s_f \in F_0$, and either $b = d$ or $b = \mathsf{anon}$, return yes. Else return no.

Figure 3: N2RPQ evaluation procedure EvalAtom.

PROCEDURE EvalQuery

Input: Boolean CN2RPQ q, $\mathcal{ELHI}_\bot$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in normal form

1. Test whether $\mathcal{K}$ is satisfiable, output yes if not.
2. Set $Q = \mathsf{Rewrite}(q, \mathcal{T})$. Replace all atoms in Q of types $C(a)$, $R(a,b)$ by equivalent atoms of type $\mathbf{A}_{s_0,F_0}(a,b)$.
3. Guess some $q' \in Q$ and an assignment $\vec{a}$ of individuals to the quantified variables $\vec{v}$ in q'
   - Let q'' be obtained by substituting $\vec{a}$ for $\vec{v}$.
   - For every atom $\mathbf{A}_{s_0,F_0}(a,b)$ in q''
     check if $\mathsf{EvalAtom}((\mathbf{A}, s_0, F_0), \mathcal{K}, (a,b)) = \mathtt{yes}$
   - If all checks succeed, return yes.
4. Return no.

Figure 4: CN2RPQ entailment procedure EvalQuery.

EvalQuery follow easily from the corresponding properties of the component procedures Rewrite and EvalAtom (Propositions 6.4, 6.5, and 6.6). In *DL-Lite$_\mathcal{R}$*, KB satisfiability is known to be NL-complete in data complexity. Since the rewriting step is ABox-independent, the size of queries in $\mathcal{Q}$ can be treated as a constant. It follows that the query q' and assignment $\vec{a}$ guessed in Step 3 can be stored in logarithmic space in $|\mathcal{A}|$. By Theorem 6.7, each call to EvalAtom runs in non-deterministic logarithmic space. □

Corollary 6.8. *CN2RPQ entailment over DL-Lite$_\mathcal{R}$ knowl-*

edge bases is NL*-complete in data complexity.*

7 Conclusions and Future Work

We have studied the extension of (C)2RPQs with a nesting construct inspired by XPath, and have characterized the data and combined complexity of answering nested 2RPQs and C2RPQs for a wide range of DLs. The only complexity bound we leave open is whether the coNP lower-bound in data complexity for expressive DLs is tight; indeed, the automata-theoretic approach used to obtain optimal bounds in combined complexity for these logics does not seem to provide the right tool for tight bounds in data complexity.

In light of the surprising jump from P to EXP in the combined complexity of answering N2RPQs in lightweight DLs, a relevant research problem is to identify classes of N2RPQs that exhibit better computational properties. We are also interested in exploring whether the techniques developed in Section 6 can be extended to deal with additional query constructs, such as existential "loop-tests" or forms of role-value maps. Finally, containment of N2RPQs has been studied very recently (Reutter 2013), but only for plain graph databases, so it would be interesting to investigate containment also in the presence of DL constraints.

Acknowledgments. This work has been partially supported by ANR project PAGODA (ANR-12-JS02-007-01), by the EU IP Project FP7-318338 *Scalable End-user Access to Big Data* (Optique), by the FWF projects T515-N23 and P25518-N23, and by the WWTF project ICT12-015.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of OWLED*.

Barceló Baeza, P. 2013. Querying graph databases. In *Proc. of PODS*, 175–188.

Barceló, P.; Libkin, L.; Lin, A. W.; and Wood, P. T. 2012. Expressive languages for path queries over graph-structured data. *ACM TODS* 37(4):31.

Berglund, A., et al. 2010. XML Path Language (XPath) 2.0 (Second Edition). W3C Recommendation. Available at `http://www.w3.org/TR/xpath20`.

Bienvenu, M.; Calvanese, D.; Ortiz, M.; and Simkus, M. 2014. Nested Regular Path Queries in Description Logics. CoRR Technical Report arXiv:1402.7122. Available at `http://arxiv.org/abs/1402.7122`.

Bienvenu, M.; Ortiz, M.; and Simkus, M. 2013. Conjunctive regular path queries in lightweight description logics. In *Proc. of IJCAI*.

Borgida, A., and Brachman, R. J. 2003. Conceptual modeling with description logics. In Baader et al. (2003). chapter 10, 349–372.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2000. Containment of conjunctive regular path queries with inverse. In *Proc. of KR*, 176–185.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2003. Reasoning on regular path queries. *SIGMOD Record* 32(4):83–92.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2006. Data complexity of query answering in description logics. In *Proc. of KR*, 260–270.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *JAR* 39(3):385–429.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2008. Conjunctive query containment and answering under description logics constraints. *ACM TOCL* 9(3):22.1–22.31.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2009. Regular path queries in expressive description logics with nominals. In *Proc. of IJCAI*, 714–720.

Chandra, A. K.; Kozen, D.; and Stockmeyer, L. J. 1981. Alternation. *JACM* 28(1):114–133.

Consens, M. P., and Mendelzon, A. O. 1990. GraphLog: a visual formalism for real life recursion. In *Proc. of PODS*, 404–416.

Cruz, I. F.; Mendelzon, A. O.; and Wood, P. T. 1987. A graphical query language supporting recursion. In *Proc. of ACM SIGMOD*, 323–330.

Eiter, T.; Lutz, C.; Ortiz, M.; and Simkus, M. 2009. Query answering in description logics with transitive roles. In *Proc. of IJCAI*, 759–764.

Glimm, B.; Lutz, C.; Horrocks, I.; and Sattler, U. 2008. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. *JAIR* 31:157–204.

Harris, S., and Seaborne, A. 2013. SPARQL 1.1 Query Language. W3C Recommendation, World Wide Web Consortium. Available at `http://www.w3.org/TR/sparql11-query`.

Hustadt, U.; Motik, B.; and Sattler, U. 2005. Data complexity of reasoning in very expressive description logics. In *Proc. of IJCAI*. 466–471

Krisnadhi, A., and Lutz, C. 2007. Data complexity in the $\mathcal{EL}$ family of description logics. In *Proc. of LPAR*, 333–347.

Lutz, C. 2008. The complexity of conjunctive query answering in expressive description logics. In *Proc. of IJCAR*, volume 5195 of *LNAI*, 179–193. Springer.

Ortiz, M.; Calvanese, D.; and Eiter, T. 2008. Data complexity of query answering in expressive description logics via tableaux. *JAR* 41(1):61–98.

Ortiz, M.; Rudolph, S.; and Simkus, M. 2011. Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In *Proc. of IJCAI*, 1039–1044.

Papadimitriou, C. H. 1993. *Computational Complexity*. Addison Wesley.

Pérez, J.; Arenas, M.; and Gutierrez, C. 2010. nSPARQL: A navigational language for RDF. *J. of Web Semantics* 8(4):255–270.

Reutter, J. L. 2013. Containment of nested regular expressions. CoRR Technical Report arXiv:1304.2637, arXiv.org e-Print archive. Available at `http://arxiv.org/abs/1304.2637`.