Journal of Artificial Intelligence Research 53 (2015) 315–374 Submitted 09/14; published 07/15

Regular Path Queries in Lightweight Description Logics: Complexity and Algorithms

Meghyn Bienvenu MEGHYN@LRI.FR
Laboratoire de Recherche en Informatique,
CNRS & Université Paris-Sud, France

Magdalena Ortiz ORTIZ@KR.TUWIEN.AC.AT
Mantas Šimkus SIMKUS@DBAI.TUWIEN.AC.AT
Institute of Information Systems,
TU Wien, Austria

Abstract

Conjunctive regular path queries are an expressive extension of the well-known class of conjunctive queries. Such queries have been extensively studied in the (graph) database community, since they support a controlled form of recursion and enable sophisticated path navigation. Somewhat surprisingly, there has been little work aimed at using such queries in the context of description logic (DL) knowledge bases, particularly for the lightweight DLs that are considered best 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. Our results demonstrate that in data complexity, the cost of moving to this richer query language is as low as one could wish for: the problem is NL-complete for DL-Lite and P-complete for $\mathcal{EL}$. The combined complexity of query answering increases from NP- to PSpace-complete, but for two-way regular path queries (without conjunction), we show that query answering is tractable even with respect to combined complexity. Our results reveal two-way conjunctive regular path queries as a promising language for querying data enriched by ontologies formulated in DLs of the DL-Lite and $\mathcal{EL}$ families or the corresponding OWL 2 QL and EL profiles.

1. Introduction

Recent years have seen a rapidly growing interest in using description logic (DL) ontologies to query instance data. This setting can be seen as a generalization of 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 & Mendelzon, 1990; Barceló, Libkin, Lin, & Wood, 2012). 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. In the presence of a DL ontology, the domain knowledge expressed in the ontology is exploited when querying the data, which can facilitate query formulation and provide users with more complete answers to their queries. While the DL and database communities share some common research goals, the research agendas they have pursued differ significantly. In the DL research, the focus has been on studying the computational complexity of answering (unions of) plain *conjunctive queries* (CQs) in the presence ontological constraints expressed in different DLs, and on the development of efficient algorithms for this setting, see the surveys of Ortiz (2013) and Ortiz and Šimkus (2012). By contrast, the work on graph databases typically does not consider ontological knowledge, but instead aims at supporting expressive *navigational query languages*.

Regular path queries (RPQs) constitute the basic navigational query language. Formally, an RPQ is given as a regular language (represented as a regular expression or a finite automaton) over the binary predicates in the database facts (arc labels, or *roles* in DL parlance), and it returns all pairs of objects that are connected by a path whose label is a word belonging to the specified language. A crucial feature of these queries is that they allow for controlled form of recursion that is computationally well behaved yet sufficient for expressing reachability queries and the traversal of paths of unbounded length (Florescu, Levy, & Suciu, 1998). In two-way RPQs (2RPQs), the regular expressions may use the arc labels in the backwards direction, which allows for more flexible path navigation. Notably, this comes at no computational cost: answering both RPQs and 2RPQs over (plain) graph databases is complete for NL in combined complexity (that is, when the complexity is measured in terms of the whole input, which in this case consists of the query and the data). *Conjunctive (two-way) regular path queries* (C(2)RPQs), which are one of the most expressive and popular languages for querying graph databases, simultaneously extend plain CQs and (2)RPQs by allowing for conjunctions of atoms that can share variables in arbitrary ways, where the atoms may contain regular expressions that navigate the arcs of the database. If we consider the data complexity measure (in which the complexity is measured only in terms of the size of the data, with all other inputs considered as fixed), then answering C2RPQs is still NL-complete. In combined complexity, the C2RPQ answering problem is NP-complete, which is in fact the lowest complexity that could be expected, given that CQ answering is already NP-hard. We note that the navigational capabilities provided by RPQs and their extensions have long been considered crucial for querying data on the Web. Indeed, navigation along regular paths is at the heart of the XPath language for querying XML data (Berglund, Boag, Chamberlin, Fernández, Kay, Robie, & Siméon, 2007), and SPARQL 1.1, the language recently recommended by the World Wide Web Consortium (W3C) as the new standard for querying RDF data (Harris & Seaborne, 2013), adds to the previous standard a feature called *property paths*, which roughly amounts to extending the 'core' of SPARQL from CQs to C2RPQs.

It comes as a surprise that, despite their advantages and relevance, RPQs and their extensions have received rather little attention in the DL literature. They were first considered in the seminal work of Calvanese, de Giacomo, and Lenzerini (1998) on query answering in the presence of DL ontologies. However, the vast majority of subsequent research has targeted *instance queries* (IQs), conjunctive queries and unions thereof, and occasionally positive first-order queries. Only a handful of works have considered C2RPQs. Calvanese et al. (2014, 2007, 2009) showed that C2RPQ answering is 2ExpTime-complete in combined complexity for the very expressive DLs[1] $\mathcal{ZIQ}$, $\mathcal{ZIO}$, and $\mathcal{ZOQ}$, which allow for regular expressions as role constructors. The same upper bound was shown for containment of C2RPQs in the presence of rich ontological knowledge (Calvanese et al., 2009). This is noteworthy since it is a well-known fact that most forms of recursion make query answer-

1. The $\mathcal{Z}$ symbol was introduced as an abbreviation for $\mathcal{ALCb}^{\mathsf{Self}}_{reg}$ (Calvanese et al., 2009).

ing and query containment undecidable in the presence of ontological constraints (Levy & Rousset, 1996). Moreover, this complexity is not higher than that of other significantly more restricted settings, such as answering plain CQs in the DL $\mathcal{ALCI}$ (Lutz, 2008) or positive first-order queries in $\mathcal{ALC}$ (Ortiz & Šimkus, 2014). However, hardness for 2ExpTime is nevertheless a prohibitively high complexity for many applications. Even in data complexity, the algorithms underlying the aforementioned results still need exponential time. More recently, algorithms for answering C2RPQs in Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ were proposed (Ortiz, Rudolph, & Šimkus, 2011). These algorithms run in polynomial time in the size of the data, but may still require exponential time in the size of the ontology, which is worst-case optimal for these logics. By contrast, for the lightweight DLs of the DL-Lite (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2007) and $\mathcal{EL}$ (Baader, Brandt, & Lutz, 2005) families, which are the languages of choice for ontology-mediated query answering and notably underlie the OWL 2 QL and EL profiles (Motik, Cuenca Grau, Horrocks, Wu, Fokoue, & Lutz, 2012), the precise complexity of answering C2RPQs was left open: the ExpTime upper bound in combined complexity for more expressive Horn DLs does not match the NP lower bound stemming of answering plain CQs, and it is not at all apparent how to obtain better upper bounds for C2RPQs by adapting existing techniques. For the DL-Lite family, the data complexity was also left open, with a gap between the NL-hardness of RPQ answering inherited from the graph database setting and the P upper bound that had been established for more expressive Horn DLs.

In this paper, we close these open questions by presenting algorithms and precise complexity bounds for answering (C)2RPQs in the $\mathcal{EL}$ and DL-Lite families of lightweight DLs. Many of our results were first announced in the conference version of this paper (Bienvenu, Ortiz, & Šimkus, 2013), but here we provide full proofs of these results, additional examples, and a discussion of extensions of our results and their applicability to the OWL 2 profiles. We also strengthen some of the complexity lower bounds by considering more restricted classes of queries (Proposition 4.5) or more succinct representations (Theorem 4.2) and identify interesting restrictions that lead to better combined complexity for C2RPQs (Theorems 6.10 and 6.11). Our main contributions can be summarized as follows:

- We establish a P lower bound in combined complexity for answering 2RPQs in DL-Lite, as well as for RPQs in DL-Lite$_\mathcal{R}$, which can be contrasted with the NL-completeness of instance checking in these logics. This result improves upon a similar lower bound from the conference version of this paper by adopting the (less succinct) regular expression representation of queries.
- We present an algorithm for answering 2RPQs over DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ knowledge bases that runs in polynomial time in combined complexity. This tractability result is extended to all single-atom C2RPQs, including those with existential variables.
- We show that answering CRPQs is PSpace-hard for both DL-Lite and $\mathcal{EL}$. This result had already been shown in the conference version, but here we provide a different proof that holds even for the structurally-restricted class of *strongly acyclic CRPQs* in which the regular expressions are disjunction-free and of star-height two. This hardness result is interesting when compared with the graph database setting, where

C2RPQ answering is NP-complete (and thus not harder than CQs in the worst case) and becomes feasible in polynomial time when restricted to strongly acyclic C2RPQs.

- We develop a query rewriting procedure for answering C2RPQs in DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$, which we use to show that the problem is feasible in PSpace for both logics. The PSpace upper bound for $\mathcal{ELH}$ is especially interesting, since C2RPQs allow for inverse roles and it is well known that adding inverse roles to $\mathcal{EL}$ immediately leads to ExpTime-hardness of query answering, even for instance queries. This result demonstrates that by including inverses in the query language, rather than the ontology language, it is possible to obtain algorithms that use only polynomial space.

- Using the same algorithm, we derive an NL upper bound in data complexity for DL-Lite$_\mathcal{R}$ and a P upper bound for $\mathcal{ELH}$. In both cases, this is the lowest data complexity that could be expected in light of existing results.

- We also identify cases where C2RPQ answering is feasible in NP, and thus not harder than answering plain CQs. This is the case for queries whose existential variables do not occur in joins (or only occur in joins in a restricted way), or for arbitrary C2RPQs whenever the ontology is guaranteed to have a finite canonical model.

Our new complexity results (and relevant existing results) are summarized in Figure 1.

This paper is organized as follows. We begin in Section 2 by introducing the lightweight description logics DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ (and relevant sublogics) and recalling some basic notions related to regular languages and computational complexity. In Section 3, we define the syntax and semantics of the different types of path queries considered in this paper. Section 4 is dedicated to showing lower bounds, first for RPQs and then for CRPQs. In both cases, we start by presenting some easy bounds that follow from known results, before moving on to the main results. Section 5 presents algorithmic techniques and upper bounds for 2RPQs, and then in Section 6, we show how these can be extended to handle C2RPQs. In Section 7, we give a brief overview of related results for similar query languages and other DLs and discuss the applicability of our results to the profiles of the OWL Web Ontology Language. Conclusions and directions for future work are given in Section 8. To improve the readability of the paper, one of the more technical proofs is deferred to the appendix.

2. Preliminaries

We briefly recall some basics of description logics, a few computational complexity classes that are relevant in the paper, and some notation for regular languages.

2.1 Description Logics

We first recall the syntax and semantics of description logics, focusing on the lightweight families of logics DL-Lite (Calvanese et al., 2007) and $\mathcal{EL}$ (Baader et al., 2005). We also recall the definition of canonical model for these logics.

	IQ		**(2)RPQ**	
	data	combined	data	combined
DL-Lite$_{\mathsf{RDFS}}$	in AC^0	NL-c	**NL-c**	**NL-c**
		$\geq$ (A)	$\geq$ (A)	$\leq$ Thm 5.2
DL-Lite$_{(\mathcal{R})}$	in AC^0	NL-c	**NL-c**	**P-c**†
	$\leq$ (G)	$\leq$ (G)	$\leq$ Thm 5.9	$\geq$ Thm 4.2, $\leq$ Thm 5.9
$\mathcal{EL}(\mathcal{H})$	P-c	P-c	**P-c**	**P-c**
	$\geq$ (E)	$\leq$ (C)	$\leq$ Thm 5.9	$\leq$ Thm 5.9

	CQ		**C(2)RPQ**	
	data	combined	data	combined
DL-Lite$_{\mathsf{RDFS}}$	in AC^0	NP-c	**NL-c**	**NP-c**
		$\geq$ (B)	$\leq$ Thm 6.8	$\leq$ Thm 6.8
DL-Lite$_{(\mathcal{R})}$	in AC^0	NP-c	**NL-c**	**PSpace-c**
	$\leq$ (D)	$\leq$ (D)	$\leq$ Thm 6.8	$\geq$ Prop 4.5, $\leq$ Thm 6.8
$\mathcal{EL}(\mathcal{H})$	P-c	NP-c	**P-c**	**PSpace-c**
	$\leq$ (F)	$\leq$ (F)	$\leq$ Thm 6.8	$\geq$ Prop 4.5, $\leq$ Thm 6.8

Figure 1: Complexity of query answering. The 'c' indicates completeness results, and '$\leq$' and '$\geq$' are used for upper and lower bounds respectively. New results are marked in bold. The remaining annotations have the following meanings:

$\dagger$	P-hardness for RPQs applies only to DL-Lite$_{\mathcal{R}}$
(A)	Easy reduction from the NL-complete directed reachability problem
(B)	Follows from NP-hardness of CQ answering over relational databases
(C)	Baader et al. (2005)
(D)	Calvanese et al. (2007)
(E)	Calvanese, De Giacomo, Lembo, Lenzerini, and Rosati (2006)
(F)	Rosati (2007), Krisnadhi and Lutz (2007), Krötzsch and Rudolph (2007)
(G)	Calvanese et al. (2007), Artale, Calvanese, Kontchakov, and Zakharyaschev (2009)

2.1.1 Description Logic Syntax

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*, respectively. An *inverse role* takes the form r^- where $r \in \mathsf{N_R}$. We will use $\mathsf{N_R^\pm}$ to refer to $\mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$, and if $R \in \mathsf{N_R^\pm}$, we use R^- to mean r^- if $R = r$ and r if $R = r^-$.

A description logic *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The former provides general domain knowledge, while the latter expresses facts about particular entities. Sometimes we use the generic terms *ontology* and *data(set)* in place of TBox and ABox.

Formally, a *TBox* is a finite set of inclusions, whose form depends on the DL in question. In DL-Lite, TBoxes consist of a set of *concept inclusions* of the form $B \sqsubseteq C$, with B and C *concepts* constructed according to the following syntax:

$$B := A \mid \exists R \qquad C := B \mid \neg B$$

where $A \in \mathsf{N_C}$ and $R \in \mathsf{N_R^{\pm}}$. DL-Lite$_{\mathcal{R}}$ additionally allows for *role inclusions* of the form $R_1 \sqsubseteq (\neg)R_2$, where $R_1, R_2 \in \mathsf{N_R^{\pm}}$. The logic DL-Lite$_{\mathsf{RDFS}}$ is obtained from DL-Lite$_{\mathcal{R}}$ by disallowing inclusions that contain negation or have existential concepts ($\exists R$) on the right-hand side. DL-Lite$_{\mathcal{R}}$ is the basis for the OWL 2 QL profile, and DL-Lite$_{\mathsf{RDFS}}$ corresponds to the fragment of DL-Lite$_{\mathcal{R}}$ that is expressible in the RDF Schema ontology language (Brickley & Guha, 2014).

In $\mathcal{EL}$, concept inclusions have the form $C_1 \sqsubseteq C_2$ where C_1, C_2 are complex concepts constructed according to the following syntax:

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

where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. The DL $\mathcal{ELH}$ additionally allows for role inclusions of the form $r_1 \sqsubseteq r_2$, where $r_1, r_2 \in \mathsf{N_R}$. Note that in $\mathcal{EL}(\mathcal{H})$ TBoxes, inverse roles are not permitted.

We will use $\mathsf{sig}(\mathcal{T})$ to denote the *signature* of a TBox $\mathcal{T}$, that is, the set of all concept and role names appearing in $\mathcal{T}$. It will also prove useful to introduce the set $\mathsf{BC}_{\mathcal{T}}$ of *basic concepts for a TBox* $\mathcal{T}$, defined as follows: $\mathsf{BC}_{\mathcal{T}} = (\mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})) \cup \{\exists r, \exists r^- \mid r \in \mathsf{N_R} \cap \mathsf{sig}(\mathcal{T})\}$ if $\mathcal{T}$ is a DL-Lite$_{\mathcal{R}}$ TBox, and $\mathsf{BC}_{\mathcal{T}} = (\mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})) \cup \{\top\}$ if $\mathcal{T}$ is an $\mathcal{ELH}$ TBox.

In all of the considered DLs, an *ABox* is a finite set of *concept assertions* of the form $A(b)$ and *role assertions* of the form $r(b, c)$, 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 appearing in the ABox $\mathcal{A}$.

2.1.2 Description Logic Semantics

The semantics of DL KBs is defined in terms of *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 individual $a \in \mathsf{N_I}$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each concept name $A \in \mathsf{N_C}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each role name $r \in \mathsf{N_R}$ to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is extended to general concepts and roles as follows:

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} & (r^-)^{\mathcal{I}} &= \{(c,d) \mid (d,c) \in r^{\mathcal{I}}\} \\ (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} & (\exists R)^{\mathcal{I}} &= \{c \mid \exists d : (c,d) \in R^{\mathcal{I}}\} \\ (\neg R)^{\mathcal{I}} &= (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}} & (\exists r.C)^{\mathcal{I}} &= \{c \mid \exists d : (c,d) \in r^{\mathcal{I}}, d \in C^{\mathcal{I}}\} \end{aligned}$$

An interpretation $\mathcal{I}$ satisfies an inclusion $G \sqsubseteq H$, denoted $\mathcal{I} \models G \sqsubseteq H$, if $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$. Similarly, $\mathcal{I}$ satisfies an assertion $A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, in symbols $\mathcal{I} \models A(a)$; $\mathcal{I}$ satisfies an assertion $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, in symbols $\mathcal{I} \models r(a, b)$. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{T}$, if $\mathcal{I}$ satisfies all inclusions in $\mathcal{T}$; it is a model of $\mathcal{A}$ if it satisfies all assertions in $\mathcal{A}$; and it is a model of $(\mathcal{T}, \mathcal{A})$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$. A KB $(\mathcal{T}, \mathcal{A})$ is *satisfiable* if it possesses at least one model, else it is unsatisfiable. Note that $\mathcal{ELH}$ knowledge bases are

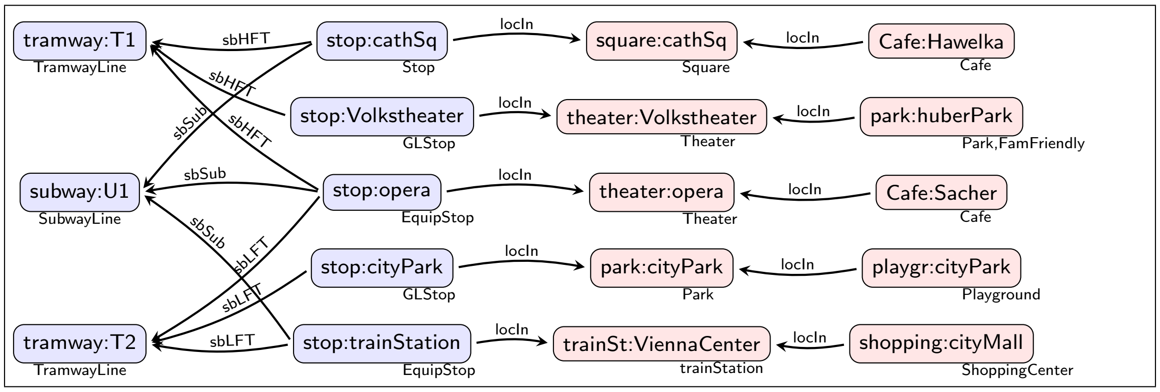

Figure 2: Example ABox $\mathcal{A}_{mob}$

always satisfiable. If $\mathcal{G}$ is a TBox, ABox, or KB, and α an inclusion or assertion, we say that $\mathcal{G}$ *entails* α, written $\mathcal{G} \models \alpha$, if $\mathcal{I} \models \alpha$ for every model $\mathcal{I}$ of $\mathcal{G}$.

Observe that we do not make the *Unique Names Assumption (UNA)*, as our definition of interpretation allows distinct individuals to be mapped to the same domain element. We remark however that all of the results in this paper hold equally well under the UNA.

Example 2.1. As a motivating example, we consider the domain of public transport and urban mobility. A partial database for this domain is presented in Figure 2. The nodes in the two leftmost columns (shaded in blue) correspond to public transport lines and stations. The arrows between them represent the existing connections, and are labeled according to the type of transport line serving them: sbSub stands for 'served by subway', while sbLFT and sbHFT respectively stand for 'served by low-floor tramway' and 'served by high-floor tramway'. Nodes are also labeled with classes in which they participate. In particular, the labels GLStop and EquipStop indicate two specific kinds of public transport stops: ground-level stops, and stops that are suitably equipped with ramps and elevators for passengers with restricted mobility. The remaining columns contain places of interest, and the locIn-labeled arrows between them represent the 'located in' relation. Note that this is nothing else but a graphical representation of a DL ABox, which we call $\mathcal{A}_{mob}$, where each label A on a node b corresponds to the concept assertion $A(b)$, and an arc labeled r from node b to node c corresponds to the role assertion $r(b, c)$.

In our example, we assume that the information in the first two columns is provided and maintained by the local public transport authorities, thus it is complete and well structured. In contrast, the remaining data is crowd-sourced, thus it is likely to be incomplete and may not adhere to a rigid, predefined structure.

In order to better respond to user queries, this data is enriched with domain knowledge expressed by the $\mathcal{EL}$ ontology in Figure 3. The ontology defines subclass relations between concepts like 'stop' and the more specialized 'accessible stop', 'ground-level stop', and it defines new terms not present in the data but which may be useful at query time, such as 'food services'. It also enhances the possibly incomplete data by asserting the existence of other places of interest. For example, a 'family-friendly' location has both dining and playground facilities.

(1) An accessible stop ($\mathsf{AccStop}$) is a public transport stop (Stop). A ground-level stop (GLStop) or a stop that is suitably equipped with ramps and elevators ($\mathsf{EquipStop}$) is an accessible stop.

(2) Restaurants and cafes are food services ($\mathsf{FoodServ}$).

(3) A place that is family friendly ($\mathsf{FamFriendly}$) has some food service and a playground.

(4) A shopping center has a supermarket and a food court.

(5) A food court has some food service.

$$\mathsf{AccStop} \sqsubseteq \mathsf{Stop} \tag{1a}$$
$$\mathsf{GLStop} \sqsubseteq \mathsf{AccStop} \tag{1b}$$
$$\mathsf{EquipStop} \sqsubseteq \mathsf{AccStop} \tag{1c}$$
$$\mathsf{Restaurant} \sqsubseteq \mathsf{FoodServ} \tag{2a}$$
$$\mathsf{Cafe} \sqsubseteq \mathsf{FoodServ} \tag{2b}$$
$$\mathsf{FamFriendly} \sqsubseteq \exists \mathsf{hasFacility}.\mathsf{FoodServ} \tag{3a}$$
$$\mathsf{FamFriendly} \sqsubseteq \exists \mathsf{hasFacility}.\mathsf{Playground} \tag{3b}$$
$$\mathsf{ShoppingCenter} \sqsubseteq \exists \mathsf{hasFacility}.\mathsf{Foodcourt} \tag{4a}$$
$$\mathsf{ShoppingCenter} \sqsubseteq \exists \mathsf{hasFacility}.\mathsf{Supermarket} \tag{4b}$$
$$\mathsf{Foodcourt} \sqsubseteq \exists \mathsf{hasFacility}.\mathsf{FoodServ} \tag{5a}$$

Figure 3: Example DL-Lite TBox $\mathcal{T}_{mob}$ expressing domain knowledge

To keep the example compact, we have chosen to write the example ontology in $\mathcal{EL}$, but the same knowledge can be expressed in DL-Lite$_{\mathcal{R}}$ by using auxiliary roles to simulate the concept inclusions with qualified existential quantification on the right-hand-side. For instance, the concept inclusion (3a) can be replaced by the following three inclusions in the syntax of DL-Lite:

$$\begin{aligned} \mathsf{FamFriendly} &\sqsubseteq \exists \mathsf{hasFoodServ} \\ \exists \mathsf{hasFoodServ}^- &\sqsubseteq \mathsf{FoodServ} \\ \mathsf{hasFoodServ} &\sqsubseteq \mathsf{hasFacility} \end{aligned}$$

and similarly for (3b), (4a), (4b), and (5a).

2.1.3 Normal Form for $\mathcal{ELH}$ TBoxes

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

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

with $A, A_1, A_2, B \in \mathsf{N_C} \cup \{\top\}$. The following well-known property (see (Baader et al., 2005)) shows that this assumption is without loss of generality.

Proposition 2.2. *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 $\mathcal{T}'$ is a* model conservative extension *of $\mathcal{T}$, that is, every model of $\mathcal{T}'$ is a model of $\mathcal{T}$, and for every model $\mathcal{I}$ of $\mathcal{T}$, there is a model $\mathcal{I}'$ of $\mathcal{T}'$ such that $\mathcal{I}$ and $\mathcal{I}'$ have the same domain and coincide on the interpretation of all concept and role names except those in $\mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$.*

2.1.4 Canonical Models

Canonical models are a key technical tool used to study lightweight description logics, and we will use them in the proofs of many of our results.

We recall the definition of the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ (alternatively denoted $\mathcal{I}_{\mathcal{K}}$) of a satisfiable DL-Lite$_{\mathcal{R}}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ consists of sequences 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 concept, and each R_i is a (possibly inverse) role. The exact definition depends on which logic we consider:

(A) When $\mathcal{T}$ is a DL-Lite$_{\mathcal{R}}$ TBox, the domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ contains exactly those sequences $aR_1\exists R_1^- \ldots R_n \exists R_n^-$ which satisfy:

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

(B) When $\mathcal{T}$ is an $\mathcal{ELH}$ TBox,[2] the domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ contains exactly those sequences $ar_1A_1 \ldots r_nA_n$ 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 $1 \leq i < n$, $\mathcal{T} \models A_i \sqsubseteq \exists r_{i+1}.A_{i+1}$.

For elements $e \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A})$, we will use the notation $\mathsf{Tail}(e)$ to denote the final concept in e. The set $\mathsf{TC}_{\mathcal{T}}$ of *tail concepts for a TBox $\mathcal{T}$* is defined as follows: $\mathsf{TC}_{\mathcal{T}} = \{\exists r, \exists r^- \mid r \in \mathsf{N_R} \cap \mathsf{sig}(\mathcal{T})\}$ if $\mathcal{T}$ is a DL-Lite$_{\mathcal{R}}$ TBox, and $\mathsf{TC}_{\mathcal{T}} = \mathsf{BC}_{\mathcal{T}} = (\mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})) \cup \{\top\}$ if $\mathcal{T}$ is an $\mathcal{ELH}$ TBox. Clearly, if $e \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A})$, then $\mathsf{Tail}(e) \in \mathsf{TC}_{\mathcal{T}}$.

To complete the definition of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, we must fix the interpretation of the individual names, concept names, and role names. This is done as follows:

$$\begin{aligned}
A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T}, \mathcal{A} \models A(a)\} \cup \ \{e \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T} \models \mathsf{Tail}(e) \sqsubseteq A\} \\
r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{(a, b) \mid \mathcal{T}, \mathcal{A} \models r(a, b)\} \cup \{(e_1, e_2) \mid e_2 = e_1 S\, C \text{ and } \mathcal{T} \models S \sqsubseteq r\} \cup \\
&\qquad \{(e_2, e_1) \mid \ e_2 = e_1 S\, C \text{ and } \mathcal{T} \models S \sqsubseteq r^-\} \\
a^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= a \qquad \text{for all } a \in \mathsf{Ind}(\mathcal{A})
\end{aligned}$$

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 the 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 domain to those elements containing e as a prefix. Observe that, by construction, $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_e$ and $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_{e'}$ are isomorphic whenever $\mathsf{Tail}(e) = \mathsf{Tail}(e')$.

It can be verified that $\mathcal{I}_{\mathcal{K}} \models \mathcal{K}$ for every satisfiable KB $\mathcal{K}$. Moreover, it is well known that the canonical model $\mathcal{I}_{\mathcal{K}}$ can be homomorphically embedded into any model of $\mathcal{K}$.

Example 2.3. The canonical model of $(\mathcal{T}_{mob}, \mathcal{A}_{mob})$ from Example 2.1 is depicted in Figure 4. To satisfy the existential restrictions in the inclusions (3a) – (5a) of $\mathcal{T}_{mob}$, the

2. Recall that throughout the paper, we assume that $\mathcal{ELH}$ KBs are in normal form, and for this reason, we only need to consider existential concepts of the form $\exists r.A$ with $A \in \mathsf{N_C} \cup \{\top\}$.

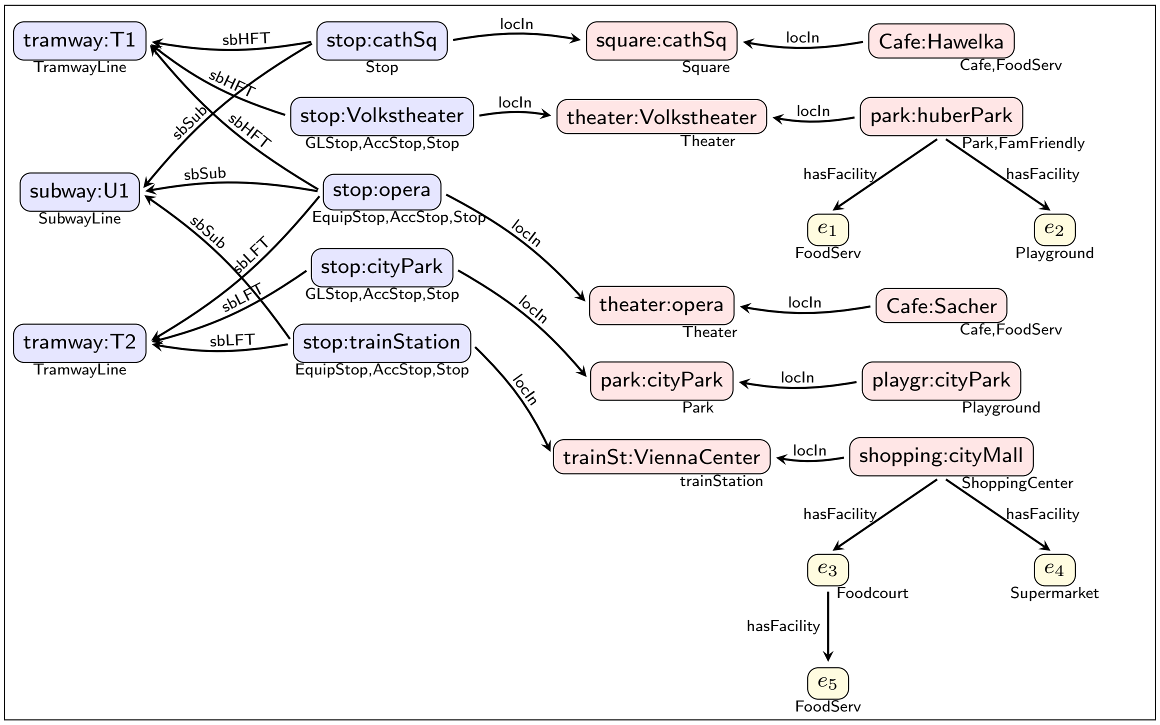

Figure 4: Canonical model $\mathcal{I}_{\mathcal{T}_{mob},\mathcal{A}_{mob}}$ of the KB $(\mathcal{T}_{mob}, \mathcal{A}_{mob})$

canonical model contains the following five anonymous elements $e_1, \ldots e_5$, which form two tree-shaped structures rooted at the nodes park:huberPark and shopping:cityMall:

$e_1 =$ park:huberPark hasFacility FoodServ

$e_2 =$ park:huberPark hasFacility Playground

$e_3 =$ shopping:cityMall hasFacility Foodcourt

$e_4 =$ shopping:cityMall hasFacility Supermarket

$e_5 =$ shopping:cityMall hasFacility Foodcourt hasFacility FoodServ

Example 2.4. To illustrate why canonical models can be infinite, we present in Figure 5 a simple DL-Lite$_\mathcal{R}$ knowledge base $(\mathcal{T}, \mathcal{A})$ and a depiction of its canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. As in the preceding example, we use names e_i as abbreviations for the anonymous objects in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. Note that the tree rooted at b_2 is infinite, since every object that belongs to the concept B has an r-child that also belongs to B.

2.2 Regular Languages

We assume the reader is familiar with regular languages, represented either by regular expressions or nondeterministic finite state automata (NFAs). *Regular expressions* $\mathcal{E}$ over alphabet Σ are defined by the grammar

$$\mathcal{E} \to \varepsilon \mid \sigma \mid \mathcal{E} \cdot \mathcal{E} \mid \mathcal{E} \cup \mathcal{E} \mid \mathcal{E}^*$$

where $\sigma \in \Sigma$ and ε denotes the empty word (i.e., the sequence of length 0). An *NFA* over an *alphabet* Σ is a tuple $\alpha = (S, \Sigma, \delta, s_0, F)$, where S is a finite set of *states*, $\delta \subseteq S \times \Sigma \times S$ the

$$\mathcal{T} = \{ \quad B \sqsubseteq \exists r, \quad \exists r^- \sqsubseteq B, \quad B \sqsubseteq \exists r_1, \quad r_1 \sqsubseteq r_2^- \quad \}$$

$$\mathcal{A} = \{ \; r(a,b), \; r_2(b,c), \; D(b) \; \}$$

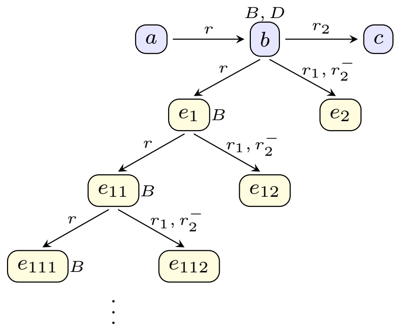

Figure 5: Example DL-Lite$_\mathcal{R}$ knowledge base $(\mathcal{T}, \mathcal{A})$ and its canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$

transition relation, $s_0 \in S$ the *initial state*, and $F \subseteq S$ the set of *final states*. We use $L(\mathcal{E})$ (resp. $L(\alpha)$) to denote the language defined by the regular expression $\mathcal{E}$ (resp. the NFA α).

We recall that NFAs are exponentially more succinct than regular expressions, in that there exists a polynomial translation of regular expressions into equivalent NFAs, while the translation from NFAs to regular expressions may incur an exponential blowup (Ehrenfeucht & Zeiger, 1974). Thus, to ensure that our complexity results hold regardless of the chosen representation, we will prove our complexity lower bounds using the regular expression representation, and for our upper bounds, we will adopt the NFA representation.

2.3 Computational Complexity

We assume familiarity with standard complexity classes, such as NL (problems solvable in non-deterministic logarithmic space), P (problems solvable in polynomial time), NP (problems solvable in non-deterministic polynomial time), PSpace (problems solvable in polynomial space), and NPspace (problems solvable in non-deterministic polynomial space). We recall that by Savitch's theorem, we have NPspace = PSpace. We shall also consider the oracle classes $\mathrm{NL}^{\mathrm{NL}}$ and NL^{P} consisting of problems solvable in non-deterministic logarithmic space when given access to an NL (respectively, P) oracle. It is well-known that $\mathrm{NL}^{\mathrm{NL}} = \mathrm{NL}$ and $\mathrm{NL}^{\mathrm{P}} = \mathrm{P}$. The circuit complexity class AC^0 that was mentioned in Figure 1 comprises problems that can be computed by a family of unbounded fan-in circuits of constant depth and polynomial size. The preceding classes are ordered as follows:

$$\mathrm{AC}^0 \subsetneq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSpace}$$

For precise definitions of these complexity classes, and other standard notions of computational complexity, we refer the reader to the recent textbook of Arora and Barak (2009) and references therein.

3. Path Queries

In this section, we introduce the different query languages considered in this paper and define the relevant computational problems.

$$q_1(x,y) = \mathsf{AccStop?} \cdot \big((\mathsf{sbSub} \cdot \mathsf{sbSub}^-) \cup (\mathsf{sbLFT} \cdot \mathsf{sbLFT}^-)\big)^* \cdot \mathsf{AccStop?}(x,y)$$

$$\begin{aligned} q_2(x,y) = \exists z_1, z_2.\ & \mathsf{AccStop?} \cdot \big((\mathsf{sbSub} \cdot \mathsf{sbSub}^-) \cup (\mathsf{sbLFT} \cdot \mathsf{sbLFT}^-)\big)^* \cdot \mathsf{AccStop?}(x,y) \\ & \wedge\ \mathsf{locIn} \cdot (\mathsf{locIn}^-)^* \cdot \mathsf{hasFacility}^* \cdot \mathsf{FoodServ?}\,(x, z_1) \\ & \wedge\ \mathsf{locIn} \cdot (\mathsf{locIn}^-)^* \cdot \mathsf{hasFacility}^* \cdot \mathsf{Playground?}\,(y, z_2) \end{aligned}$$

Figure 6: Example queries

3.1 Syntax of Path Queries

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 disjoint tuples of variables, and φ is a conjunction of atoms of the forms:

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

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

As usual, variables and individuals are called *terms*, the variables in $\vec{x}$ are called *answer variables*, and the variables in $\vec{y}$ are called *quantified variables.* We use $\mathsf{terms}(q)$, $\mathsf{vars}(q)$, $\mathsf{avars}(q)$, and $\mathsf{qvars}(q)$ to refer respectively to the sets of terms, variables, answer variables, and quantified variables appearing in query q. A query with no answer variables is called a *Boolean query.* Note that where convenient we will treat a query as its set of atoms.

Conjunctive (one-way) regular path queries (CRPQs) are obtained by disallowing symbols from $\mathsf{N_R^\pm} \setminus \mathsf{N_R}$ 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) such that t and t' are both answer variables. Regular path queries (RPQs)* are 2RPQs that do not use any symbols from $\mathsf{N_R^\pm} \setminus \mathsf{N_R}$. Finally, *instance queries (IQs)* take the form $A(x)$ with $A \in \mathsf{N_C}$, or $r(x, y)$ with $r \in \mathsf{N_R}$.

Note that it will sometimes prove convenient to treat queries as sets of atoms, using the notation $\alpha \in q$ to indicate that α is an atom of q.

Example 3.1. Figure 6 shows two example queries. The 2RPQ q_1 retrieves pairs x, y of public transport stops that have an accessible connection, that is, both x and y are accessible stops, and there is a public transport route between them that uses only subway and low-floor tramway. The query q_2 retrieves pairs x, y of public transport stops that have an accessible connection (as in q_1) and such that there is a place to eat at the location of x and a playground at the location of y.

Note that by using the Kleene star (*), we can query for services such as restaurants and playgrounds available at some location without having to take care of the different ways in which places can be related. For example, a restaurant could be at the same location as a stop, or it could be that it is in a food court inside some shopping center that is itself at the same location as a stop. In both cases, q_2 correctly identifies it as a food service at that location. This is a very useful feature of RPQs and their extensions, particularly in cases where the data does not comply to a rigid schema.

3.2 Semantics of Path Queries

We now proceed to define the semantics of C2RPQs. Given an interpretation $\mathcal{I}$, a *path* from e_0 to e_n in $\mathcal{I}$ is a sequence $e_0 u_1 e_1 u_2 \ldots u_n e_n$ with $n \geq 0$ such that every e_i is an element from $\Delta^{\mathcal{I}}$, every u_i is a symbol from $\mathsf{N}_\mathsf{R}^{\pm} \cup \{A? \mid A \in \mathsf{N}_\mathsf{C}\}$, and for every $1 \leq i \leq n$:

- If $u_i = A?$, then $e_{i-1} = e_i \in A^{\mathcal{I}}$;
- If $u_i = R \in \mathsf{N}_\mathsf{R}^{\pm}$, then $(e_{i-1}, e_i) \in R^{\mathcal{I}}$.

The *label* $\lambda(p)$ of path $p = e_0 u_1 e_1 u_2 \ldots u_n e_n$ is the word $u_1 u_2 \ldots u_n$. Note that if $p = e_0$, then we define $\lambda(p)$ to be ε.

Then for every language L over $\mathsf{N}_\mathsf{R}^{\pm} \cup \{A? \mid A \in \mathsf{N}_\mathsf{C}\}$, the semantics of L w.r.t. interpretation $\mathcal{I}$ is defined as follows:

$$L^{\mathcal{I}} = \{(e_0, e_n) \mid \text{there is some path } p \text{ from } e_0 \text{ to } e_n \text{ with } \lambda(p) \in L\}$$

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

- $\pi(c) = c^{\mathcal{I}}$ for each $c \in \mathsf{N}_\mathsf{I}$,
- $\pi(t) \in A^{\mathcal{I}}$ for each atom $A(t)$ in q, and
- $(\pi(t), \pi(t')) \in L(\Lambda)^{\mathcal{I}}$ for each $\Lambda(t, t')$ in q.

Given a C2RPQ q with answer variables $x_1, \ldots, x_k$, we say that a tuple of individuals $(a_1, \ldots, a_k)$ from $\mathsf{Ind}(\mathcal{A})$ is a *certain answer* to q w.r.t. the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ just in the case that in every model $\mathcal{I}$ of $\mathcal{K}$ there is a match π for q such that $\pi(v_i) = a_i^{\mathcal{I}}$ for every $1 \leq i \leq k$. We use $\mathsf{cert}(q, \mathcal{K})$ to denote the set of certain answers to q w.r.t. the KB $\mathcal{K}$. Note that if q is a Boolean query, then either $\mathsf{cert}(q, \mathcal{K}) = \{()\}$ (where $()$ denotes the empty tuple) or $\mathsf{cert}(q, \mathcal{K}) = \emptyset$. In the former case, we say that q is entailed from $\mathcal{K}$, and we write $\mathcal{K} \models q$.

We remark that the normal form for $\mathcal{ELH}$ TBoxes can also be assumed without loss of generality for query answering. Indeed, we can always assume that the fresh symbols in the TBox $\mathcal{T}'$ in normal form do not occur in q, and it follows from Proposition 2.2 and the definition of certain answers that $\mathsf{cert}(q, (\mathcal{T}, \mathcal{A})) = \mathsf{cert}(q, (\mathcal{T}', \mathcal{A}))$ for every C2RPQ q that does not use any symbols in $\mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$.

While the definition of certain answers involves all models of a KB, for the DLs considered in this paper, it is in fact sufficient to look for matches in the canonical model.

Lemma 3.2. *For every DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, C2RPQ $q(\vec{x})$ of arity k, and k-tuple $\vec{a}$ of individuals from $\mathcal{A}$: $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$ if and only if there is a match π for q in $\mathcal{I}_\mathcal{K}$ such that $\pi(\vec{x}) = \vec{a}$.*

Proof sketch. It is well known that the canonical model $\mathcal{I}_\mathcal{K}$ can be homomorphically embedded into any model of $\mathcal{K}$ (Calvanese et al., 2007; Rosati, 2007; Krisnadhi & Lutz, 2007; Krötzsch & Rudolph, 2007). It follows that whenever query matches are preserved under homomorphisms, the existence of a match in $\mathcal{I}_\mathcal{K}$ implies the existence of a match in every model. This has been often observed for CQs, and it applies equally well to C2RPQs (Calvanese et al., 2014; Ortiz et al., 2011). Since the converse is trivially true, certain answers coincide with the answers over the canonical model. □

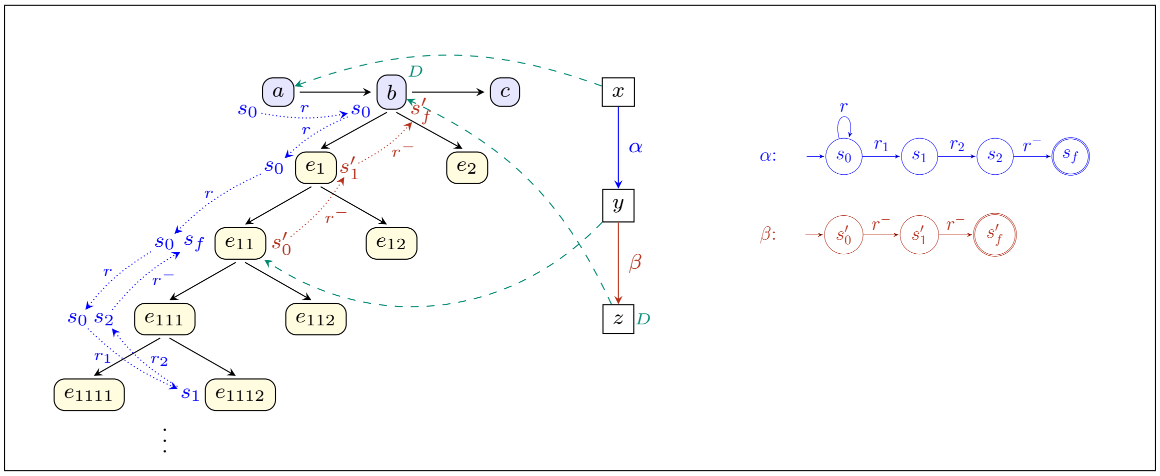

Figure 7: A match witnessing $a \in \mathsf{cert}(q, \mathcal{K})$ for q and $\mathcal{K}$ from Example 3.4

Example 3.3. In our urban mobility example, the stop at the cathedral square is not known to be accessible (i.e., $\mathsf{AccStop}(\mathsf{stop{:}cathSq})$ is not entailed by $(\mathcal{T}_{mob}, \mathcal{A}_{mob})$), hence $\mathsf{stop{:}cathSq}$ cannot participate in any match for q_1. The stop at the theater is accessible, but it is only connected to other stops via high-floor tramway. Thus $\mathsf{stop{:}Volkstheater}$ only participates in one mapping for q_1 in $\mathcal{I}_{\mathcal{T}_{mob},\mathcal{A}_{mob}}$, namely $\pi(x) = \pi(y) = \mathsf{stop{:}Volkstheater}$. Indeed, the path

$$\mathsf{stop{:}Volkstheater}\ \mathsf{AccStop?}\ \mathsf{stop{:}Volkstheater}\ \mathsf{AccStop?}\ \mathsf{stop{:}Volkstheater}$$

witnesses that $(\mathsf{stop{:}Volkstheater}, \mathsf{stop{:}Volkstheater}) \in L_1^{\mathcal{I}}$ for the language L_1 specified in q_1, and there is no longer path starting or ending at $\mathsf{stop{:}Volkstheater}$ whose label belongs to L_1. The stops $\mathsf{stop{:}opera}$, $\mathsf{stop{:}cityPark}$, and $\mathsf{stop{:}trainStation}$ are all accessible and mutually connected via accessible public transport (i.e., subway and low-floor tramway lines). Hence, we can find a path between any pair of them whose label is in L_1, and all such pairs are certain answers to q_1. Thus $\mathsf{cert}(q_1, (\mathcal{T}_{mob}, \mathcal{A}_{mob}))$ contains $(\mathsf{stop{:}Volkstheater}, \mathsf{stop{:}Volkstheater})$, and all pairs of stops involving $\mathsf{stop{:}opera}$, $\mathsf{stop{:}cityPark}$, and $\mathsf{stop{:}trainStation}$.

The certain answers to q_2 are precisely those pairs (s_1, s_2) of stops that are an answer to q_1 such that there is some food service at the location of s_1 and a playground at the location of s_2. Since in $\mathcal{I}_{\mathcal{T}_{mob},\mathcal{A}_{mob}}$, we find both some food service and a playground at the location of $\mathsf{stop{:}Volkstheater}$, we have $(\mathsf{stop{:}Volkstheater}, \mathsf{stop{:}Volkstheater}) \in \mathsf{cert}(q_2, (\mathcal{T}_{mob}, \mathcal{A}_{mob}))$. We also find food services at the locations of $\mathsf{stop{:}opera}$ and $\mathsf{stop{:}trainStation}$, and a playground at the location of $\mathsf{stop{:}cityPark}$, hence $\mathsf{cert}(q_2, (\mathcal{T}_{mob}, \mathcal{A}_{mob}))$ also contains the pairs $(\mathsf{stop{:}opera}, \mathsf{stop{:}cityPark})$ and $(\mathsf{stop{:}trainStation}, \mathsf{stop{:}cityPark})$.

Example 3.4. We also give an example of a query over the KB $(\mathcal{T}, \mathcal{A})$ in Figure 5:

$$q(x) = \exists y, z.\ r^* \cdot r_1 \cdot r_2 \cdot r^-(x, y) \wedge r^- \cdot r^-(y, z) \wedge D(z)$$

We have $\mathsf{cert}(q, \mathcal{K}) = \{a, b\}$. To see why a is a certain answer, consider the mapping $\pi(x) = a$, $\pi(y) = e_{11}$, and $\pi(z) = b$. The path $arbre_1re_{11}re_{111}r_1e_{1112}r_2e_{111}r^-e_{11}$ witnesses that $(a, e_{11}) \in L(r^* \cdot r_1 \cdot r_2 \cdot r^-)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, while the path $e_{11}r^-e_1r^-b$ witnesses that $(e_{11}, b) \in$

$L(r^- \cdot r^-)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Since $b \in D^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, the mapping π is a match for q. This match is depicted in Figure 7. To see that b is also a certain answer, consider $\pi'(x) = \pi'(z) = b$ and $\pi'(y) = e_{11}$, and observe that π' is also a match because $bre_1re_{11}re_{111}r_1e_{1112}r_2e_{111}r^-e_{11}$ witnesses $(b, e_{11}) \in L(r^* \cdot r_1 \cdot r_2 \cdot r^-)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

Note that in both matches, y is mapped to an element in the anonymous part, and there is no match mapping y to an individual. This illustrates that anonymous elements may play a decisive role in query answering, and every complete query answering algorithm must consider possible matches into the possibly infinite anonymous part of canonical models.

3.3 Computational Problems

In this paper, we will be interested in the problem of computing the certain answers to C2RPQs, and more precisely, the associated decision problem of determining whether a given tuple is a certain answer to a query. In what follows, for a query language $Q \in$ {IQ, CQ, RPQ, CRPQ, 2RPQ, C2RPQ}, we will use the term Q *answering* to refer to the problem of deciding given a KB $\mathcal{K}$, tuple $\vec{a}$, and query q from Q, whether $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$.

There are different ways of measuring the complexity of query answering, depending on which of the three parameters of the problem ($\mathcal{T}$, $\mathcal{A}$, and q) are considered as inputs and which are considered fixed. In this work, we consider the two most commonly used measures: combined complexity and data complexity. *Combined complexity* treats all three parameters as inputs, so the complexity is measured with respect to the total size $|\mathcal{T}|+|\mathcal{A}|+|q|$ (we use $|\cdot|$ to denote the *size* of an object, e.g. the length of its string representation according to some suitable encoding). *Data complexity* takes $\mathcal{A}$ as input and assumes $\mathcal{T}$ and q to be fixed, so the complexity is measured only with respect to $|\mathcal{A}|$, with $|\mathcal{T}|$ and $|q|$ treated as constants.

4. Lower Bounds

In this section, we establish the required complexity lower bounds. We begin with some lower bounds for RPQs that can be straightforwardly obtained from existing results.

Proposition 4.1. *RPQ answering is*

1. NL-*hard in data complexity for DL-Lite*$_{\mathsf{RDFS}}$*;*
2. P-*hard in data complexity for* $\mathcal{EL}$*;*

Proof. Statement (1) follows from the analogous result for graph databases (Consens & Mendelzon, 1990). It can be shown by a simple reduction from the NL-complete directed reachability problem: vertex b is reachable from vertex a in a directed graph G if and only if (a, b) is a certain answer to the RPQ $r^*(x, y)$ w.r.t. the KB $(\emptyset, \mathcal{A}_G)$, where

$$\mathcal{A}_G = \{r(v_1, v_2) \mid \text{there is a directed edge from } v_1 \text{ to } v_2 \text{ in } G\}.$$

Statement (2) is a direct consequence of the P-hardness in data complexity of instance checking in $\mathcal{EL}$ (Calvanese et al., 2006), since the instance query $A(x)$ can be computed using the RPQ $A?(x, y)$. □

In the case of DL-Lite, we establish a P lower bound for 2RPQs, which contrasts with the NL-completeness of instance checking. We remark that a similar result was given in the conference version of this paper (Bienvenu et al., 2013), but the reduction required an NFA representation of the regular language in the query. The complexity of 2RPQs using the (less succinct) regular expression representation was left open and is resolved by the following theorem.

Theorem 4.2. *2RPQ answering is P-hard in combined complexity for DL-Lite.*

Proof. We give a reduction from the P-complete entailment problem for propositional definite Horn theories. Without loss of generality, we suppose that we are given a propositional Horn theory Ψ over variables $v_1, \ldots, v_n$ that consists of

- a set of rules $\rho_i = v_{i_1} \wedge v_{i_2} \to v_{i_3}$ $(1 \leq i \leq m)$
- a single "initialization" fact: v_1, with $v_1 \neq v_n$

Indeed, any arbitrary propositional definite Horn theory Ψ' can be transformed into a theory of the preceding form as follows: take a fresh variable v_1 not appearing in Ψ', add v_1 to the body of every rule in Ψ', add the fact v_1, and finally perform standard syntactic manipulations (possibly introducing additional fresh variables) to ensure that all rules other than the initialization fact v_1 contain exactly two body variables.

In what follows, we show how to construct, given a propositional Horn theory Ψ of the form above, a DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and Boolean 2RPQ q such that $\mathcal{K} \models q$ if and only if $\Psi \models v_n$. We first provide an informal description of the reduction, and then present it formally. As is well known, $\Psi \models v_n$ just in the case that there exists a *proof tree* for v_n from Ψ, which can be defined as a binary tree T where each node is labeled with a variable from $v_1, \ldots, v_n$ such that the following conditions are satisfied: (i) the root is labeled with v_n, (ii) the leaves are labeled with v_1, and (iii) for any inner node d, if d is labeled with v_k, then there is a rule $\rho_i \in \Psi$ such that $v_k = v_{i_3}$ and the two children of d are labeled with v_{i_1} and v_{i_2}, respectively. The existence of a node-labeled proof tree T as just described is equivalent to the existence of an *edge-labeled proof tree* T', defined as a sibling-ordered binary tree whose edges are labeled with rules from Ψ as follows. First, the two edges outgoing from the root are labeled with a rule ρ_i with $v_{i_3} = v_n$. For a non-root node d_ℓ which is the ℓ-th child of its parent d, where $\ell \in \{1, 2\}$, if the edge (d, d_ℓ) is labeled with ρ_i, then either $v_{i_\ell} = v_1$, or d_ℓ has two outgoing edges E_1 and E_2 that are both labeled with a rule ρ_j such that $v_{j_3} = v_{i_\ell}$.

We will next show how to construct $\mathcal{K}$ and q such that $\mathcal{K} \models q$ if and only if there exists a edge-labeled proof tree T' for v_n from Ψ. Roughly speaking, we use $\mathcal{K}$ to generate in the anonymous part a tree that contains all possible edge-labeled proof trees. Since such proof trees are based upon sibling-ordered trees, we need to distinguish between the first and second children of a node, and so we use two roles $r_{i,1}$ and $r_{i,2}$ for each rule ρ_i. An edge-labeled proof tree T' will thus map into a subtree of $\mathcal{I}_\mathcal{K}$ of the same structure, but with label ρ_i replaced by either $r_{i,1}$ or $r_{i,2}$, depending on whether the edge leads to the first or second child of the parent node. We then use a 2RPQ q to determine whether $\mathcal{I}_\mathcal{K}$ actually contains such a subtree. The intuition is that every path that witnesses the satisfaction of q corresponds to the complete depth-first traversal of (the representation of) a valid edge-labeled proof tree that starts and ends at the root, and in which the left subtree of a node is always visited before the right one.

$$\mathcal{E} = (\bigcup_{1\le i\le m} r_{i,1}) \cdot \left(\bigcup_{1\le i\le m} r_{i,1} \cup \bigcup_{k\in F_1} (r^-_{k,1} \cdot r_{k,2}) \cup \bigcup_{k\in F_2} (r^-_{k,2} \cdot (\bigcup_{1\le i\le m} r^-_{i,2})^* \cdot (\bigcup_{1\le i\le m} (r^-_{i,1} \cdot r_{i,2}) \cup \varepsilon)) \right)^*$$

Figure 8: Regular expression used in the proof of Theorem 4.2.

The ABox $\mathcal{A}$ consists of a single assertion $A(a)$, and the TBox $\mathcal{T}$ contains the following concept inclusions:

- $A \sqsubseteq \exists r_{i,\ell}$, where $\ell \in \{1,2\}$ and $1 \le i \le m$ with $v_{i_3} = v_n$
- $\exists r^-_{i,\ell} \sqsubseteq \exists r_{k,j}$, where $\ell, j \in \{1,2\}$ and $1 \le i, k \le m$ such that $v_{i_\ell} = v_{k_3}$

For $\ell \in \{1,2\}$, we define the set

$$F_\ell = \{k \mid 1 \le k \le m, v_{k_\ell} = v_1\}.$$

Intuitively, F_ℓ contains the index of each rule where we can 'turn back' since its ℓ-th variable is the initial variable v_1, (and so the corresponding child node in the proof tree would be a leaf). We use F_1 and F_2 to define the regular expression $\mathcal{E}$ in Figure 8, which we then use to define the following 2RPQ:

$$q = \mathcal{E}(a, a).$$

We now prove the correctness of the reduction.

($\Rightarrow$) Suppose that $\mathcal{K} \models q$. Then by Lemma 3.2, there is a match for q in the canonical model $\mathcal{I}_\mathcal{K}$ of $\mathcal{K}$. This means that $(a, a) = (a^{\mathcal{I}_\mathcal{K}}, a^{\mathcal{I}_\mathcal{K}}) \in L(\mathcal{E})^{\mathcal{I}_\mathcal{K}}$, and so there exists a path $e_0\sigma_1 \dots \sigma_p e_p$ in $\mathcal{I}_\mathcal{K}$ whose label is in $L(\mathcal{E})$ such that $e_0 = e_p = a$.

Claim 1. For every $2 \le j \le p$:

1. If $\sigma_j = r_{i,1}$, then there exists $j' > j$ with $\sigma_{j'} = r^-_{i,2}$.
2. If $\sigma_j = r^-_{i,2}$, then there exists $j' < j$ such that $\sigma_{j'} = r^-_{i,1}$.
3. If $\sigma_j = r^-_{i,1}$ and $i \notin F_1$, then $\sigma_{j-1} = r^-_{i',2}$ where $v_{i'_3} = v_{i_1}$.
4. If $\sigma_j = r^-_{i,2}$ and $i \notin F_2$, then $\sigma_{j-1} = r^-_{i',2}$ where $v_{i'_3} = v_{i_2}$.

Proof of claim. For Point 1, we remark that in $\mathcal{I}_\mathcal{K}$ all roles are directed away from the ABox; formally, for every role name s, if $(g, g') \in s^{\mathcal{I}_\mathcal{K}}$, then $g' = gs\exists s^-$. It follows that if $\sigma_j = r_{i,1}$, then $e_j = e_{j-1}r_{i,1}\exists r^-_{i,1}$. Since the sequence of elements $e_j, \dots, e_p$ defines a path in $\mathcal{I}_\mathcal{K}$ from e_j to $e_p = a$, by continuity, there must be some $j_0 > j$ such that $e_{j_0-1} = e_j$ and $e_{j_0} = e_{j-1}$, in which case we must have $\sigma_{j_0} = r^-_{i,1}$. Next note that the structure of $\mathcal{E}$ ensures that every occurrence of $r^-_{i,1}$ is immediately followed by $r_{i,2}$. Then repeating the same argument, substituting $r_{i,2}$ for $r_{i,1}$, we can find some $j' > j_0$ such that $\sigma_{j'} = r^-_{i,2}$.

For Point 2, we again use the fact that roles in $\mathcal{I}_\mathcal{K}$ are directed away from the ABox. Thus, if $\sigma_j = r^-_{i,2}$, then $e_{j-1} = e_j r_{i,2}\exists r^-_{i,2}$. The sequence $e_0, \dots, e_{j-1}$ of elements forms a path in $\mathcal{I}_\mathcal{K}$ from $e_0 = a$ to $e_j r_{i,2} \exists r^-_{i,2}$. Thus, by continuity, there must exist some $j' < j$ such that $e_{j'} = e_j$, $e_{j'+1} = e_j r_{i,2} \exists r^-_{i,2}$, and $\sigma_{j'+1} = r_{i,2}$. By examining the structure of $\mathcal{E}$, we can see that any occurrence of $r_{i,2}$ must be immediately preceded by $r^-_{i,1}$, and so $\sigma_{j'} = r^-_{i,1}$.

To show Point 3, suppose that $\sigma_j = r^-_{i,1}$ and $i \notin F_1$. The structure of $\mathcal{E}$ ensures that the preceding symbol σ_{j-1} takes the form $r^-_{i',2}$. We thus have $(e_j, e_{j-1}) \in r^{\mathcal{I}_\mathcal{K}}_{i,1}$ and $(e_{j-1}, e_{j-2}) \in r^{\mathcal{I}_\mathcal{K}}_{i',2}$. Once again using the fact that elements in $\mathcal{I}_\mathcal{K}$ do not contain inverse role names, we obtain $e_{j-2} = e_j r_{i,1} \exists r^-_{i,1} r_{i',2} \exists r^-_{i',2}$. It follows that $\mathcal{T} \models \exists r^-_{i,1} \sqsubseteq \exists r_{i',2}$, which can only be the case if $v_{i'_3} = v_{i_1}$.

Finally, for Point 4, suppose that $\sigma_j = r^-_{i,2}$ and $i \notin F_2$. Examining the structure of $\mathcal{E}$, it is clear that the preceding symbol σ_{j-1} must be of the form $r^-_{i',2}$, and so we have $(e_j, e_{j-1}) \in r^{\mathcal{I}_\mathcal{K}}_{i,2}$ and $(e_{j-1}, e_{j-2}) \in r^{\mathcal{I}_\mathcal{K}}_{i',2}$. This means that $\mathcal{T} \models \exists r^-_{i,2} \sqsubseteq \exists r_{i',2}$, and hence $v_{i'_3} = v_{i_2}$. (*end proof of claim*)

It is easy to see that the first symbol σ_1 must have the form $r_{i,1}$, and so we have $(e_0, e_1) \in r^{\mathcal{I}_\mathcal{K}}_{i,1}$. Since $e_0 = a$, we have $(a, a r_{i,1} \exists r^-_{i,1}) \in r^{\mathcal{I}_\mathcal{K}}_{i,1}$. This implies that $\mathcal{T} \models A \sqsubseteq \exists r_{i,1}$, and hence that $v_{i_3} = v_n$. Applying Points 1 and 2 of the preceding claim, we can find j, k such that $\sigma_j = r^-_{i,1}$ and $\sigma_k = r^-_{i,2}$. To complete the proof of this direction, we establish the following claim.

Claim 2. For every $1 \leq j \leq p$ and $\ell \in \{1, 2\}$, if $\sigma_j = r^-_{i,\ell}$, then $\Psi \models v_{i_\ell}$.

Proof of claim. We proceed by induction on j. For the base case, suppose that $\sigma_j = r^-_{i,\ell}$, and there is no $j' < j$ with $\sigma_{j'} = r^-_{i',\ell'}$. It follows from Claim 1 that $i \in F_1$ and $\ell = 1$. We thus have $v_{i_\ell} = v_1$, so $\Psi \models v_{i_\ell}$ trivially holds.

For the induction step, suppose that claim holds for all $j < k$, and consider $\sigma_k = r^-_{i,\ell}$. We consider three cases:

- Case 1: $\sigma_k = r^-_{i,\ell}$ and $i \in F_\ell$
  Since $i \in F_\ell$, we have $v_{i_\ell} = v_1$, so $\Psi \models v_{i_\ell}$ follows immediately.

- Case 2: $\sigma_k = r^-_{i,1}$ and $i \notin F_1$
  By Point 3 of Claim 1, $\sigma_{k-1} = r^-_{i',2}$ where $v_{i'_3} = v_{i_1}$. Applying the induction hypothesis to σ_{k-1}, we obtain $\Psi \models v_{i'_2}$. From Point 2 of Claim 2, there exists some $j < k$ such that $\sigma_j = r^-_{i',1}$. A second application of the induction hypothesis yields $\Psi \models v_{i'_1}$. The rule $v_{i'_1} \wedge v_{i'_2} \rightarrow v_{i'_3}$ belongs to Ψ, so we must also have $\Psi \models v_{i'_3}$. Then since $v_{i'_3} = v_{i_1}$, we obtain $\Psi \models v_{i_1}$.

- Case 3: $\sigma_k = r^-_{i,2}$ and $i \notin F_2$
  We can use almost the same argument as for Case 2, except that we must use Point 4 of Claim 1, rather than Point 3. (*end proof of claim*)

As Ψ contains the rule $v_{i_1} \wedge v_{i_2} \rightarrow v_{i_3}$, it follows from Claim 2 that $\Psi \models v_n$.

($\Leftarrow$) If $\Psi \models v_n$, then there must be exist an edge-labeled proof tree T' for v_n from Ψ as described at the beginning of the proof. We define a mapping f from the nodes of T' to domain elements in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ as follows:

- $f(d) = a$ for d the root of T';
- for every non-root node d, if d is the first (resp., second) child of its parent d_p and (d_p, d) is labeled ρ_i, then $f(d) = f(d_p)r_{i,1}\exists r^-_{i,1}$ (resp. $f(d) = f(d_p)r_{i,2}\exists r^-_{i,2}$).

We show the following claim:

Claim 3: For every non-leaf node d in T', $(f(d), f(d)) \in L(\mathcal{E})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

Proof of claim. For every non-leaf node d, we consider a depth-first traversal of the subtree of T' rooted at d that always visits the left subtree of a node before visiting the right one, and then returns to the root. Let $d_1, d_2, \dots, d_n$ be the sequence of nodes visited in this traversal, with $d_1 = d_n = d$, and for every $2 \leq i \leq n$, let $\sigma_{i-1,i}$ be as follows:

- $\sigma_{i-1,i} = r_{j,1}$ if d_i is the first child of d_{i-1} and (d_{i-1}, d_i) is labeled with ρ_j;
- $\sigma_{i-1,i} = r_{j,2}$ if d_i is the second child of d_{i-1} and (d_{i-1}, d_i) is labeled with ρ_j;
- $\sigma_{i-1,i} = r^-_{j,1}$ if d_{i-1} is the first child of d_i and (d_i, d_{i-1}) is labeled with ρ_j;
- $\sigma_{i-1,i} = r^-_{j,2}$ if d_{i-1} is the second child of d_i and (d_i, d_{i-1}) is labeled with ρ_j.

We now define a path p_d as follows:

$$p_d = f(d_1)\sigma_{1,2}f(d_2)\sigma_{2,3}\dots\sigma_{n-1,n}f(d_n)$$

To show the claim, it suffices to show the following for every non-leaf node d:

$(*)$ p_d is a path in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ with $\lambda(p_d) \in L(\mathcal{E})$.

This can be shown by induction on the minimal distance of d to a leaf in T'. An important observation is that, for $\ell \in \{1, 2\}$, if the ℓth child d^ℓ of a node d of T' is a leaf, and the edge from d to d^ℓ is labeled ρ_i, then by definition $i \in F_\ell$.

In order to be able to more easily argue that some words belong to $L(\mathcal{E})$, we give names to the relevant subexpressions of $\mathcal{E}$:

$$\mathcal{E}_1 = \bigcup_{1\leq i\leq m} r_{i,1}$$

$$\mathcal{E}_2 = \bigcup_{k\in F_1} (r^-_{k,1} \cdot r_{k,2})$$

$$\mathcal{E}_3 = (\bigcup_{1\leq i\leq m} r^-_{i,2})^*$$

$$\mathcal{E}_4 = (\bigcup_{1\leq i\leq m} (r^-_{i,1} \cdot r_{i,2}) \cup \varepsilon)$$

$$\mathcal{E}_5 = \bigcup_{k\in F_2} (r^-_{k,2} \cdot (\bigcup_{1\leq i\leq m} r^-_{i,2})^* \cdot (\bigcup_{1\leq i\leq m} (r^-_{i,1} \cdot r_{i,2}) \cup \varepsilon)$$

$$\mathcal{E}_6 = \left(\bigcup_{1\leq i\leq m} r_{i,1} \cup \bigcup_{k\in F_1} (r^-_{k,1} \cdot r_{k,2}) \cup \bigcup_{k\in F_2} (r^-_{k,2} \cdot (\bigcup_{1\leq i\leq m} r^-_{i,2})^* \cdot (\bigcup_{1\leq i\leq m} (r^-_{i,1} \cdot r_{i,2}) \cup \varepsilon))\right)^*$$

Note that we have $\mathcal{E}_5 = \bigcup_{k \in F_2}(r_{k,2}^- \cdot \mathcal{E}_3 \cdot \mathcal{E}_4)$, $\mathcal{E}_6 = (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_5)^*$, and $\mathcal{E} = \mathcal{E}_1 \cdot \mathcal{E}_6$.

Now we are ready to prove $(*)$. For the base case, when both children of d are leaves, let ρ_i be the label of the edges from d to its children. By construction of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, $f(d)$ has an $r_{i,1}$-child and an $r_{i,2}$-child, so p_d is a path in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ with $\lambda(p_d) = r_{i,1} \cdot r_{i,1}^- \cdot r_{i,2} \cdot r_{i,2}^-$. Both children of d are leaves, so $i \in F_1 \cap F_2$. We thus have $r_{i,1} \in L(\mathcal{E}_1)$, $r_{i,1}^- \cdot r_{i,2} \in L(\mathcal{E}_2)$, and $r_{i,2}^- \in L(\mathcal{E}_5) = L(\bigcup_{k \in F_2}(r_{k,2}^- \cdot \mathcal{E}_3 \cdot \mathcal{E}_4))$ (for the latter, observe that $\varepsilon \in L(\mathcal{E}_3)$ and $\varepsilon \in L(\mathcal{E}_4)$). Using $\mathcal{E}_6 = (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_5)^*$, we get $r_{i,1}^- \cdot r_{i,2} \cdot r_{i,2}^- \in L(\mathcal{E}_6)$, and using $\mathcal{E} = \mathcal{E}_1 \cdot \mathcal{E}_6$, we obtain $\lambda(p_d) = r_{i,1} \cdot r_{i,1}^- \cdot r_{i,2} \cdot r_{i,2}^- \in L(\mathcal{E})$.

For the induction step, let d^L and d^R be the left and right children of d respectively, and let ρ_i be the rule that is used to label the edges from d to d^L and from d to d^R. By construction of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ and f, we know that $f(d^L)$ is an $r_{i,1}$-child of $f(d)$ and that $f(d^R)$ is an $r_{i,2}$-child of $f(d)$. We distinguish three cases:

- If neither of the children d^L and d^R of d is a leaf, then we know from the induction hypothesis that p_{d^L} and p_{d^R} are paths in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ which start and end at $f(d^L)$ and $f(d^R)$ respectively and are such that $\{\lambda(p_{d^L}), \lambda(p_{d^R})\} \subseteq L(\mathcal{E})$. It follows that $p_d = f(d) r_{i,1} p_{d^L} r_{i,1}^- f(d) r_{i,2} p_{d^R} r_{i,2}^- f(d)$ is a path in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. We let ρ_{k^L} (resp. ρ_{k^R}) be the label linking d^L (resp. d^R) to its two children. Note that by construction, $\lambda(p_{d^L})$ (resp. $\lambda(p_{d^L})$) ends with $r_{k^L,1}^-$ (resp. $r_{k^L,2}^-$). It follows that $\lambda(p_{d^L}) \in L(\mathcal{E}_1 \cdot \mathcal{E}_6 \cdot \mathcal{E}_5)$ and that for the subexpression $\mathcal{E}_4 = (\bigcup_{1 \leq i \leq m}(r_{i,1}^- \cdot r_{i,2}) \cup \varepsilon)$ of the final $\mathcal{E}_5$, we must select ε. By choosing to instantiate $\mathcal{E}_4$ with $r_{i,1}^- r_{i,2}$ instead, we can show that $\lambda(p_{d^L}) \cdot r_{i,1}^- \cdot r_{i,2} \in L(\mathcal{E}_6)$. A similar argument for $\lambda(p_{d^R})$ can be used to show that $\lambda(p_{d^R}) \cdot r_{i,2}^- \in L(\mathcal{E}_6)$. We therefore obtain $\lambda(p_d) = r_{i,1} \cdot \lambda(p_{d_1}) \cdot r_{i,1}^- \cdot r_{i,2} \cdot \lambda(p_{d_1}) \cdot r_{i,2}^- \in L(\mathcal{E}_1 \cdot \mathcal{E}_6 \cdot \mathcal{E}_6) \subseteq L(\mathcal{E})$.
- If d^L is not a leaf but d^R is, then we can apply the induction hypothesis to infer that p_{d^L} is a path in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ that starts and ends at $f(d^L)$ and is such that $\lambda(p_{d^L}) \in L(\mathcal{E})$. By using the same reasoning as in the previous case, we obtain that $\lambda(p_{d^L}) \cdot r_{i,1}^- \cdot r_{i,2} \in L(\mathcal{E}_6)$. As d^R is a leaf, it follows that $i \in F_2$, and hence $r_{i,2}^- \in L(\mathcal{E}_5)$ (here again we choose ε to satisfy the subexpressions $\mathcal{E}_3$ and $\mathcal{E}_4$ of $\mathcal{E}_5$). Putting this together, we find that $\lambda(p_d) = r_{i,1} \lambda(p_{d_1}) r_{i,1}^- r_{i,2} r_{i,2}^- \in L(\mathcal{E}_1 \cdot \mathcal{E}_6 \cdot \mathcal{E}_5) \subseteq L(\mathcal{E})$.
- If d^L is a leaf but d^R is not, the argument is analogous to the previous case.

(*end proof of claim*)

Since $f(d) = a$ for the root d of T', it follows from the preceding claim that $(a, a) \in L(\mathcal{E})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, hence $\mathcal{K} \models q$. □

In DL-Lite$_\mathcal{R}$, we can strengthen Theorem 4.2 by using role inclusions to eliminate inverse roles in the query.

Corollary 4.3. *RPQ answering is* P*-hard in combined complexity in DL-Lite$_\mathcal{R}$.*

Proof. Let q be a 2RPQ and $(\mathcal{T}, \mathcal{A})$ be a DL-Lite$_\mathcal{R}$ KB. For each inverse role r^- appearing in q, we introduce a new role name r_{inv}. We then let q' be the RPQ obtained by replacing every occurrence of r^- in the query by r_{inv}, and let $\mathcal{T}'$ be the extension of $\mathcal{T}$ with the role inclusions $r^- \sqsubseteq r_{\mathsf{inv}}$ and $r_{\mathsf{inv}} \sqsubseteq r^-$ for each new role name r_{inv}. It is easy to see that $\mathsf{cert}(q, (\mathcal{T}, \mathcal{A})) = \mathsf{cert}(q', (\mathcal{T}', \mathcal{A}))$. □

We leave open whether RPQ answering in DL-Lite is P-hard in combined complexity.

We next provide combined complexity lower bounds for CRPQs. As CRPQs generalize CQs, we inherit an NP lower bound from the well-known NP-hardness in combined complexity of CQ answering in relational databases (see (Abiteboul, Hull, & Vianu, 1995)):

Proposition 4.4. *CRPQ answering is* NP*-hard in combined complexity for DL-Lite*$_{\mathsf{RDFS}}$.

For DL-Lite and $\mathcal{EL}$, we show that CRPQ answering is PSpace-hard for combined complexity, in contrast to CQ answering which is NP-complete. Interestingly, PSpace-hardness holds even under strong restrictions. In particular, we consider the following restrictions on the shape of the query and the form of the regular languages in query atoms:

- *(Strong) acyclicity*: a C2RPQ q is acyclic if its associated undirected graph $G_q = \{\{t,t'\} \mid L(t,t') \in q\}$ is acyclic. It is strongly acyclic if additionally (i) it does not contain any atoms of the form $L(t,t)$ and (ii) for every pair of distinct atoms $L_1(t_1,t_1'), L_2(t_2,t_2')$, we have $\{t_1,t_1'\} \neq \{t_2,t_2'\}$.
- *Disjunction-freeness and star-height of regular expressions*: a regular expression is disjunction-free if it does not contain $\cup$. The star-height of a regular expression is defined as the maximum nesting depth of stars appearing in the regular expression.

We point out that in the graph database setting, (strong) acyclicity leads to tractability: acyclic C2RPQs can be evaluated in polynomial time in combined complexity (Barceló et al., 2012), and for strongly C2RPQs, evaluation can even be done in linear time in the size of the database (Barceló, 2013). By contrast, the following result shows that strong acyclicity has no impact on the worst-case complexity of CRPQ answering in our setting:

Theorem 4.5. *CRPQ answering is* PSpace*-hard in combined complexity for DL-Lite and $\mathcal{EL}$. This result applies even under the restriction to strongly acyclic CRPQs whose regular languages are given by disjunction-free regular expressions of star-height two.*

Proof. 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-hard. This result was first shown by Kozen (1977) for regular languages given as deterministic NFAs. More recently, Bala (2002) proved that this problem is also PSpace-hard when the regular languages are given as disjunction-free regular expressions of star-height two. So, let $\mathcal{E}_1, \ldots, \mathcal{E}_n$ be disjunction-free regular expressions of star-height two over the alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$. We will use the symbols in Σ as role names, and we will also use concept names A and B. For our reduction, we consider the following Boolean CRPQ:

$$q = \exists x_1, \ldots, x_n, y.\, A(x_1) \wedge \ldots A(x_n) \wedge \mathcal{E}_1(x_1, y) \wedge \ldots \wedge \mathcal{E}_n(x_n, y)$$

Observe that q satisfies the restrictions in the proposition statement. For the KB, we use the ABox $\mathcal{A} = \{A(a)\}$ and a TBox $\mathcal{T}$ whose form depends on the logic in question. For DL-Lite, we use

$$\mathcal{T} = \{A \sqsubseteq \exists \sigma_i \mid \sigma_i \in \Sigma\} \cup \{\exists \sigma_i^- \sqsubseteq \exists \sigma_j \mid \sigma_i, \sigma_j \in \Sigma\}$$

and for $\mathcal{EL}$, we use instead:

$$\mathcal{T} = \{A \sqsubseteq B\} \cup \{B \sqsubseteq \exists \sigma_i.B \mid \sigma_i \in \Sigma\}.$$

Notice that in both cases the canonical model $\mathcal{I}_{\mathcal{K}}$ of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of an infinite tree rooted at a such that every element in the interpretation has a unique σ_i-child for each $\sigma_i \in \Sigma$ (and no other children). Thus, we can associate to every domain element in $\mathcal{I}_{\mathcal{K}}$ the word over Σ given by the sequence of role names encountered along the unique path from a, and moreover, for every word $w \in \Sigma^*$ we can find an element e_w such that the sequence of role names on the path from a to e_w is exactly w.

We claim that $L(\mathcal{E}_1) \cap \ldots \cap L(\mathcal{E}_n)$ is non-empty if and only if $\mathcal{K} \models q$. To see why, first note that if $w \in L(\mathcal{E}_1) \cap \ldots \cap L(\mathcal{E}_n)$, then we can define a match for q in the canonical model by mapping the variables $x_1, \ldots, x_n$ to a and y to e_w. Conversely, if q is entailed from $\mathcal{K}$, then there is a match π of q in $\mathcal{I}_{\mathcal{K}}$. Since $A^{\mathcal{I}_{\mathcal{K}}} = \{a\}$, we must have $\pi(x_i) = a$ for every $1 \leq i \leq n$. It follows that the unique path from a to $\pi(y)$ in $\mathcal{I}_{\mathcal{K}}$ is a word that belongs to every $L(\mathcal{E}_i)$, which means that $L(\mathcal{E}_1) \cap \ldots \cap L(\mathcal{E}_n)$ is non-empty. ☐

We note that the proof of the preceding theorem from the conference version (Bienvenu et al., 2013) uses a simpler query in which the variables x_i are replaced by a and the atoms $A(x_i)$ are dropped. However, that query is not strongly acyclic. We also remark that a similar proof had already been used to establish PSpace hardness of CQs in an extension of $\mathcal{ELH}$ that allows for *regular* role hierarchies (Krötzsch & Rudolph, 2007).

5. Upper Bounds for 2RPQs

In the next two sections, we provide concrete query answering algorithms for the considered classes of path queries and DLs, which we leverage to derive matching upper bounds to the complexity lower bounds from Section 4. The technical developments are presented in stages. We begin this section by giving a simple algorithm for answering 2RPQs in DL-Lite$_{\mathsf{RDFS}}$. The remainder of the section is then devoted to showing how this algorithm can be extended to handle DL-Lite$_{\mathcal{R}}$ and $\mathcal{ELH}$. Afterwards, in Section 6, we introduce a query rewriting procedure that, when combined with the algorithms from the present section, yields a method for answering C2RPQs.

In this and the following section, we assume that all binary atoms take the form $\alpha(t, t')$, where α is an NFA over $\mathsf{N}_{\mathsf{R}}^{\pm} \cup \{A? \mid A \in \mathsf{N}_{\mathsf{C}}\}$. This is without loss of generality, since every regular expression can transformed into an equivalent NFA; an examination of the standard technique for constructing NFAs from regular expressions (Thompson, 1968) reveals that the transformation can in fact be performed by a deterministic logspace transducer.

It will also be useful to introduce notation for NFAs that result from changing the initial and final states of some other NFA. In what follows, given an NFA $\alpha = (S, \Sigma, \delta, s_0, F)$, we will use $\alpha_{s,G}$ to denote the NFA $(S, \Sigma, \delta, s, G)$, i.e., the NFA with the same states and transitions as α but with initial state s and final states G. When there is a single final state s', we write $\alpha_{s,s'}$ in place of $\alpha_{s,\{s'\}}$.

```
ALGORITHM BasicEval
INPUT: NFA $\alpha = (S, \Sigma, \delta, s_0, F)$ with $\Sigma \subseteq \mathsf{N}_\mathsf{R}^\pm \cup \{A? \mid A \in \mathsf{N}_\mathsf{C}\}$, DL-Lite$_\mathsf{RDFS}$ KB $(\mathcal{T}, \mathcal{A})$,
       $(a, b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A})$

1. Initialize $\mathsf{current} = (a, s_0)$ and $\mathsf{count} = 0$. Set $\mathsf{max} = |\mathcal{A}| \cdot |S|$.

2. While $\mathsf{count} < \mathsf{max}$ and $\mathsf{current} \notin \{(b, s_f) \mid s_f \in F\}$

   (a) Let $\mathsf{current} = (c, s)$.
   (b) Guess a pair $(d, s') \in \mathsf{Ind}(\mathcal{A}) \times S$ and a transition $(s, \sigma, s') \in \delta$
       • If $\sigma \in \mathsf{N}_\mathsf{R}^\pm$, then verify that $\mathcal{T}, \mathcal{A} \models \sigma(c, d)$, and return no if not.
       • If $\sigma = A?$, then verify that $c = d$ and $\mathcal{T}, \mathcal{A} \models A(c)$, and return no if not.
   (c) Set $\mathsf{current} = (d, s')$ and increment $\mathsf{count}$.

3. If $\mathsf{current} = (b, s_f)$ for some $s_f \in F$, return yes. Else return no.
```

Figure 9: Non-deterministic algorithm for 2RPQ answering in DL-Lite$_\mathsf{RDFS}$.

5.1 Warm-up: 2RPQ Answering in DL-Lite$_\mathsf{RDFS}$

A standard technique for answering 2RPQs in the absence of an ontology is to non-deterministically guess a path between a pair of individuals that is labeled by a word from the specified regular language. Such a procedure can be made to run in logarithmic space by only keeping a small portion of the path in memory at any time.

In Figure 9, we present a simple non-deterministic algorithm BasicEval for answering 2RPQs over DL-Lite$_\mathsf{RDFS}$ knowledge bases that implements this idea. The algorithm takes as input an NFA $\alpha = (S, \Sigma, \delta, s_0, F)$, a DL-Lite$_\mathsf{RDFS}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and a pair of individuals (a, b) from $\mathcal{A}$, and it decides whether $(a, b) \in \mathsf{cert}(\alpha(x, y), \mathcal{K})$. In Step 1, we initialize current with the pair (a, s_0) and the counter count to 0. We also compute the maximum value max of the counter, which corresponds to the largest length of path that needs to be considered. At every iteration of the while loop (Step 2), we start with a single pair (c, s) stored in current and then proceed to guess a new pair (d, s') together with a transition of the form (s, σ, s'). The idea is that we would like to append σd to the path guessed so far, but to do so, we must ensure that the conditions of paths are satisfied. This is the purpose of the entailment checks in Step 2(b). If the applicable check succeeds, then we place (d, s') in current and increment count. We exit the while loop once we have reached the maximum counter value or the pair in count takes the form (b, s_f) with s_f a final state. In the latter case, we have managed to guess a path with the required properties, and so the algorithm returns yes.

The use of the counter ensures that the algorithm terminates, and the following proposition proves that it always outputs the correct result.

Proposition 5.1. *For every 2RPQ $q = \alpha(x, y)$, DL-Lite$_\mathsf{RDFS}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and pair of individuals (a, b) from $\mathsf{Ind}(\mathcal{A})$: $(a, b) \in \mathsf{cert}(q, \mathcal{K})$ if and only if there is some execution of* BasicEval$(\alpha, \mathcal{K}, (a, b))$ *that returns* yes.

Proof. Consider a 2RPQ $q = \alpha(x, y)$ with $\alpha = (S, \Sigma, \delta, s_0, F)$, a DL-Lite$_{\mathsf{RDFS}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and a pair of individuals (a, b) from $\mathsf{Ind}(\mathcal{A})$.

First suppose that $(a, b) \in \mathsf{cert}(q, \mathcal{K})$. Then there is a path $p = e_0u_1 \ldots u_ne_n$ in $\mathcal{I}_\mathcal{K}$ such that $e_0 = a$, $e_n = b$, and $\lambda(p) \in L(\alpha)$. We may assume without loss of generality that there is no path of shorter length than p that satisfies these conditions. As $\mathcal{T}$ is a DL-Lite$_{\mathsf{RDFS}}$ TBox, we know that $\Delta^{\mathcal{I}_\mathcal{K}} = \mathsf{Ind}(\mathcal{A})$, so every $e_i \in \mathsf{Ind}(\mathcal{A})$. Since $\lambda(p) \in L(\alpha)$, we can find a sequence of states $s_0s_1 \ldots s_n$ from S such that $s_n \in F$ and for every $1 \leq i \leq n$, $(s_{i-1}, u_i, s_i) \in \delta$. Because of our minimality assumption, we know that $(e_i, s_i) \neq (e_j, s_j)$ for $i \neq j$, so the sequence of pairs $(e_0, s_0)(e_1, s_1) \ldots (e_n, s_n)$ has length at most $|\mathcal{A}| \cdot |S|$. It is easily verified that by guessing this sequence of pairs, together with the corresponding transitions (s_{i-1}, u_i, s_i), we obtain an execution of $\mathsf{BasicEval}$ that returns $\mathtt{yes}$.

For the other direction, suppose that there is some execution of $\mathsf{BasicEval}(\alpha, \mathcal{K}, (a, b))$ that returns $\mathtt{yes}$, and let $(c_0, s_0)(a_1, s_1) \ldots (c_n, s_n)$ be the sequence of pairs that were guessed by this execution. Then we must have $c_0 = a$, $c_n = b$, and $s_n \in F$. Moreover, for every $1 \leq i \leq n$, there must exist a transition $(s_{i-1}, u_i, s_i) \in \delta$ such that $\mathcal{T}, \mathcal{A} \models u_i(c_{i-1}, c_i)$ if $u_i \in \mathsf{N}_\mathsf{R}^{\pm}$, or $c_{i-1} = c_i$ and $\mathcal{T}, \mathcal{A} \models A(c_i)$ if $u_i = A?$. It follows that the sequence $p = c_0u_1c_1 \ldots u_nc_n$ is a path in $\mathcal{I}_\mathcal{K}$ with $\lambda(p) \in L(\alpha)$, and so $(c_0, c_n) = (a, b) \in \mathsf{cert}(q, \mathcal{K})$. □

By analyzing the complexity of the procedure $\mathsf{BasicEval}$, we obtain an NL upper bound, which matches the NL lower bound from Proposition 4.1.

Theorem 5.2. *2RPQ answering is in* NL *in combined complexity for DL-Lite*$_{\mathsf{RDFS}}$.

Proof. By Proposition 5.1, we know that $\mathsf{BasicEval}$ is a decision procedure for 2RPQ answering in DL-Lite$_{\mathsf{RDFS}}$. To see why $\mathsf{BasicEval}$ runs in non-deterministic logarithmic space, we note that (i) by utilizing a binary encoding, only logarithmic space is needed to store the value of count, the old and new values of $\mathsf{current}$, and the guessed transition from δ, and (ii) the entailment checks in 2(b) can be performed in non-deterministic logarithmic space. The latter follows from the fact that instance checking is in NL in combined complexity in the superlogic DL-Lite$_\mathcal{R}$ (Calvanese et al., 2007). □

5.2 2RPQ Answering in DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$

We now turn to the problem of answering 2RPQs over DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ knowledge bases. The following example shows that the basic evaluation algorithm we used for DL-Lite$_{\mathsf{RDFS}}$ is incomplete for these logics. Intuitively, the problem lies in the fact that the algorithm only considers paths along ABox individuals, whereas to satisfy the query it may be necessary to consider paths that pass through the anonymous part.

Example 5.3. In the remaining of the paper, we will consider the KB $(\mathcal{T}, \mathcal{A})$ with $\mathcal{T} = \{B \sqsubseteq \exists r, \exists r^- \sqsubseteq B, B \sqsubseteq \exists r_1, r_1 \sqsubseteq r_2^-\}$ and $\mathcal{A} = \{r(a, b), r_2(b, c), D(b)\}$ from Example 2.4. As an example 2RPQ, we consider $q'(x, y) = \alpha(x, y)$, where $\alpha = \langle S, \Sigma, \delta, s_0, \{s_f\}\rangle$ with

$$\begin{aligned} S &= \{s_0, s_1, s_2, s_f\} \\ \Sigma &= \{r, r_1, r_2, r^-\} \\ \delta &= \{(s_0, r, s_0), (s_0, r_1, s_1), (s_1, r_2, s_2), (s_2, r^-, s_f)\} \end{aligned}$$

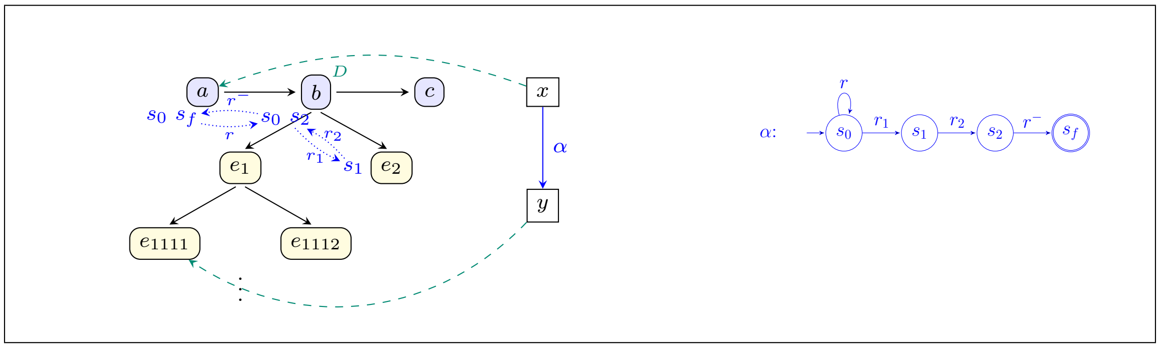

Figure 10: A match witnessing $(a, a) \in \mathsf{cert}(q, \mathcal{K})$ for $q(x, y) = \alpha(x, y)$

The NFA α is depicted in the upper right-hand-side of Figure 10. Observe that $L(\alpha) = L(r^* \cdot r_1 \cdot r_2 \cdot r^-)$ (i.e., the same language as in the first atom of the C2RPQ q of Example 3.4). Note that $(a, a) \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$, but this is only witnessed by the path $arbr_1e_2r_2br^-a$, which passes by the element e_2 in the anonymous part of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. On this input, the algorithm $\mathsf{BasicEval}$ would start from (a, s_0). In the first iteration of the while loop, it could guess the pair (b, s_0) and the transition (s_0, r, s_0), and the entailment check in the first item of Step 2(b) would succeed since $\mathcal{T}, \mathcal{A} \models r(a, b)$. However, in the next iteration the only transitions from s_0 are (s_0, r, s_0), (s_0, r_1, s_1), but there is no $d \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{T}, \mathcal{A} \models r(b, d)$ or $\mathcal{T}, \mathcal{A} \models r_1(b, d)$. Hence there is no good pair to guess in Step 2(b) and the algorithm would fail. Since there are no other possible guesses in the first iteration that satisfy the entailment check, the algorithm incorrectly returns `no`.

Our aim is to modify the evaluation algorithm to take into account detours through the anonymous part. We observe that any path between two ABox individuals in $\mathcal{I}_{\mathcal{K}}$ can be decomposed into a sequence of paths of two types:

- paths whose elements all belong to the ABox
- paths that begin and end with the same ABox individual and whose intermediate points all belong to the anonymous part

Paths of the first type are already handled by the evaluation algorithm. To handle paths of the second type, we will show how to check whether there is a path of this form that starts and ends at a given individual a and takes the query automaton from state s to state s'. If such a “loop” exists at the individual a, then during query evaluation, we will be allowed to jump directly from (a, s) to (a, s'). By modifying the evaluation algorithm to allow such shortcuts in addition to normal transitions, we can ensure that all possible paths in the canonical model are taken into account.

A key observation is that to decide whether a loop is available at a given ABox individual a it is sufficient to consider the basic concepts that hold at a. This leads us to define a table ALoop_α in which the entry $\mathsf{ALoop}_\alpha[s, s']$ contains the set of all basic concepts C that force the existence of a path of the second type whose label takes the query automaton from state s to state s'. In order to define ALoop_α, we will require a second table Loop_α that will contain for each pair of states (s, s'), a set of tail concepts that guarantee the existence of a path from an anonymous element e to itself that takes the query automaton from s to s' while never leaving the subtree of $\mathcal{I}_\mathcal{K}$ rooted at e (note that we do allow e to occur multiple times along the path).

Let us now proceed to the definition of the tables ALoop_α and Loop_α. If $\mathcal{T}$ is a DL-Lite$_\mathcal{R}$ TBox, then Loop_α is defined inductively using the following rules:

(L1) For every $s \in S$: $\mathsf{Loop}_\alpha[s,s] = \mathsf{TC}_\mathcal{T}$.

(L2) If $C \in \mathsf{Loop}_\alpha[s_1,s_2]$ and $C \in \mathsf{Loop}_\alpha[s_2,s_3]$, then $C \in \mathsf{Loop}_\alpha[s_1,s_3]$.

(L3) If $C \in \mathsf{TC}_\mathcal{T}$, $\mathcal{T} \models C \sqsubseteq A$, and $(s_1, A?, s_2) \in \delta$, then $C \in \mathsf{Loop}_\alpha[s_1,s_2]$.

(L4) If $C \in \mathsf{TC}_\mathcal{T}$, $\mathcal{T} \models C \sqsubseteq \exists R$, $\mathcal{T} \models R \sqsubseteq R'$, $\mathcal{T} \models R^- \sqsubseteq R''$, $(s_1, R', s_2) \in \delta$, $\exists R^- \in \mathsf{Loop}_\alpha[s_2,s_3]$, and $(s_3, R'', s_4) \in \delta$, then $C \in \mathsf{Loop}_\alpha[s_1,s_4]$.

and the table ALoop_α is constructed from Loop_α using the rule:

(L5) If $C \in \mathsf{BC}_\mathcal{T}$, $\mathcal{T} \models C \sqsubseteq \exists R$, $\mathcal{T} \models R \sqsubseteq R'$, $\mathcal{T} \models R^- \sqsubseteq R''$, $(s_1, R', s_2) \in \delta$, $\exists R^- \in \mathsf{Loop}_\alpha[s_2,s_3]$, and $(s_3, R'', s_4) \in \delta$, then $C \in \mathsf{ALoop}_\alpha[s_1,s_4]$.

For $\mathcal{ELH}$, we use the same definitions, except that the rules 4 and 5 are replaced by:

(L4') If $C \in \mathsf{TC}_\mathcal{T}$, $\mathcal{T} \models C \sqsubseteq \exists r.D$, $\mathcal{T} \models r \sqsubseteq r'$, $\mathcal{T} \models r \sqsubseteq r''$, $(s_1, r', s_2) \in \delta$, $D \in \mathsf{Loop}_\alpha[s_2,s_3]$, and $(s_3, r''^-, s_4) \in \delta$, then $C \in \mathsf{Loop}_\alpha[s_1,s_4]$.

(L5') If $C \in \mathsf{BC}_\mathcal{T}$, $\mathcal{T} \models C \sqsubseteq \exists r.D$, $\mathcal{T} \models r \sqsubseteq r'$, $\mathcal{T} \models r \sqsubseteq r''$, $(s_1, r', s_2) \in \delta$, $D \in \mathsf{Loop}_\alpha[s_2,s_3]$, and $(s_3, r''^-, s_4) \in \delta$, then $C \in \mathsf{ALoop}_\alpha[s_1,s_4]$.

Note that since $\mathsf{TC}_\mathcal{T} = \mathsf{BC}_\mathcal{T}$ in $\mathcal{ELH}$, the only difference between (L4') and (L5') is that the former adds concepts to the table Loop_α, while the latter adds concepts to ALoop_α.

The following example illustrates the construction of the tables Loop_α and ALoop_α.

Example 5.4. Observe that in our running example $\mathsf{TC}_\mathcal{T} = \{\exists r, \exists r^-, \exists r_1, \exists r_1^-, \exists r_2, \exists r_2^-\}$ and $\mathsf{BC}_\mathcal{T} = \{B\} \cup \mathsf{TC}_\mathcal{T}$. In the first step of the loop computation for the 2RPQ q' from Example 5.3, we get $\mathsf{Loop}_\alpha[s,s] = \mathsf{TC}_\mathcal{T}$ for every $s \in \{s_0, s_1, s_2, s_f\}$. We can infer that $\exists r^- \in \mathsf{Loop}_\alpha[s_0,s_2]$ using rule L4 and the following facts:

$$\begin{array}{lll} \mathcal{T} \models \exists r^- \sqsubseteq \exists r_1 & \mathcal{T} \models r_1 \sqsubseteq r_1 & \mathcal{T} \models r_1^- \sqsubseteq r_2 \\ (s_0, r_1, s_1) \in \delta & \exists r_1^- \in \mathsf{Loop}_\alpha[s_1,s_1] & (s_1, r_2, s_2) \in \delta \end{array}$$

Intuitively, every element e that satisfies $\exists r^-$ has an r_1-child e' (by $\mathcal{T} \models \exists r^- \sqsubseteq \exists r_1$), and e' is in turn an r_2-child of e (by $r_1^- \sqsubseteq r_2$). Hence, starting from such an element e, we can always use the transition (s_0, r_1, s_1) to go to the child e' and then return to e using the transition (s_1, r_2, s_2), i.e., there is a loop from s_0 to s_2 at e.

In a further step, we can infer $\exists r^- \in \mathsf{Loop}_\alpha[s_0,s_3]$ by using rule L4 together with:

$$\begin{array}{lll} \mathcal{T} \models \exists r^- \sqsubseteq \exists r & \mathcal{T} \models r \sqsubseteq r & \mathcal{T} \models r^- \sqsubseteq r^- \\ (s_0, r, s_0) \in \delta & \exists r^- \in \mathsf{Loop}_\alpha[s_0,s_2] & (s_2, r^-, s_f) \in \delta \end{array}$$

This reflects that whenever an element satisfies $\exists r^-$, there is a loop from s_0 to s_3 as follows: we can move to some r-child (which exists by $\mathcal{T} \models \exists r^- \sqsubseteq \exists r$) staying in state s_0 with (s_0, r, s_0), then use the previously computed loop to jump to s_2 at the same element, and then go back up to e with the transition (s_2, r^-, s_f).

One can verify that no further loops can be inferred with the rules, so we obtain:

$$\mathsf{Loop}_\alpha[s_i, s_i] = \mathsf{TC}_\mathcal{T} \text{ for } 0 \leq i \leq 3 \qquad \mathsf{Loop}_\alpha[s_0, s_2] = \{\exists r^-\} \qquad \mathsf{Loop}_\alpha[s_0, s_3] = \{\exists r^-\}$$

Now we compute ALoop_α. First we note that there are applications of L5 analogous to the two described applications of L4, which respectively result in $\exists r^- \in \mathsf{ALoop}_\alpha[s_0, s_2]$ and $\exists r^- \in \mathsf{ALoop}_\alpha[s_0, s_3]$. Moreover, similar applications but using $\mathcal{T} \models B \sqsubseteq \exists r_1$ instead of $\mathcal{T} \models \exists r^- \sqsubseteq \exists r_1$ and $\mathcal{T} \models B \sqsubseteq \exists r$ instead of $\mathcal{T} \models \exists r^- \sqsubseteq \exists r$ yield $B \in \mathsf{ALoop}_\alpha[s_0, s_2]$ and $B \in \mathsf{ALoop}_\alpha[s_0, s_3]$. Further applications of L5 yield no new loops, hence we obtain:

$$\mathsf{ALoop}_\alpha[s_0, s_2] = \{\exists r^-, B\} \qquad \mathsf{ALoop}_\alpha[s_0, s_3] = \{\exists r^-, B\}$$

We observe that the tables Loop_α and ALoop_α can be constructed in polynomial time in $|\mathcal{T}|$ and $|\alpha|$ since entailment of inclusions is in P for both DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ (Calvanese et al., 2007; Baader et al., 2005). The following propositions show that Loop_α and ALoop_α have the desired meaning:

Proposition 5.5. *For every DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $d \in \Delta^{\mathcal{I}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$, the following are equivalent:*

1. $\mathsf{Tail}(d) \in \mathsf{Loop}_\alpha[s, s']$;
2. *There is a path $p = e_0 u_1 e_1 \dots u_n e_n$ in $\mathcal{I}_\mathcal{K}$ such that $\lambda(p) \in L(\alpha_{s,s'})$, $e_0 = e_n = d$, and $e_i \in \Delta^{\mathcal{I}_\mathcal{K}|_d}$ for every $0 \leq i \leq n$.*

Proof. Consider a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and an automaton $\alpha = (S, \Sigma, \delta, s, F)$. We begin by proving that the first statement implies the second. Fix a sequence of applications of the rules L1, L2, L3, and L4 (or L4') which generates the full table Loop_α, and let k be the length of this sequence. It then suffices to show the following claim for all $1 \leq i \leq k$:

Claim: If C is inserted into $\mathsf{Loop}_\alpha[s, s']$ on the i-th rule application and $d \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A})$ is such that $\mathsf{Tail}(d) = C$, then there is a path $p = e_0 u_1 e_1 \dots u_n e_n$ in $\mathcal{I}_\mathcal{K}$ such that $\lambda(p) \in L(\alpha_{s,s'})$, $e_0 = e_n = d$, and $e_i \in \Delta^{\mathcal{I}_\mathcal{K}|_d}$ for every $0 \leq i \leq n$.

Proof of claim. The proof is by induction on i. First suppose that C is inserted into $\mathsf{Loop}_\alpha[s, s']$ with the first rule application and $d \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A})$ is such that $\mathsf{Tail}(d) = C$. Then either rule L1 or rule L3 must have been applied. In the former case, we have $s = s'$, so the path $p = d$ satisfies the required conditions (recall that in this case $\lambda(p) \in \varepsilon$). If instead it was rule L3 that was applied, then we must have $\mathcal{T} \models C \sqsubseteq A$ and $(s, A?, s') \in \delta$ for some concept name A. Since $\mathsf{Tail}(d) = C$, we must have $d \in A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. It follows that $p = dA?d$ is a path satisfying the required conditions.

For the induction step, suppose that the statement holds for all $1 \leq i < k$, and let $d \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A})$ be such that $C = \mathsf{Tail}(d)$ is inserted into $\mathsf{Loop}_\alpha[s, s']$ on the k-th rule application. The first possibility is that the k-th rule application involves rules L1 or L3, in which case we proceed as in the base case. The next possibility is that rule L2 was applied. Then there must exist s'' such that after the first $k-1$ rule applications, we have $C \in \mathsf{Loop}_\alpha[s,s'']$ and $C \in \mathsf{Loop}_\alpha[s'',s']$. Applying the induction hypothesis, we find paths p_1 and p_2 that both begin and end with d, contain only elements from $\Delta^{\mathcal{I}_\mathcal{K}|_d}$, and are such that $\lambda(p_1) \in L(\alpha_{s,s''})$ and $\lambda(p_2) \in L(\alpha_{s'',s'})$. Let p_3 be the path obtained by taking p_1 then adding p_2 with its first occurrence of d removed. Then p_3 begins and ends at d, contains only elements from $\Delta^{\mathcal{I}_\mathcal{K}|_d}$, and is such that $\lambda(p_3) \in L(\alpha_{s,s'})$.

The final possibility is that the k-th rule application involves rule L4. Here the proof differs depending on whether $\mathcal{T}$ is formulated in DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$. We give the proof only for DL-Lite$_\mathcal{R}$; the proof for $\mathcal{ELH}$ proceeds analogously. We first note that since an application of rule L4 leads to the insertion of C into $\mathsf{Loop}_\alpha[s,s']$ at stage k, it must be the case that we can find $R, R', R'' \in \mathsf{N}_\mathsf{R}^\pm$ and $s'', s''' \in S$ such that

- $\mathcal{T} \models C \sqsubseteq \exists R$, $\mathcal{T} \models R \sqsubseteq R'$, $\mathcal{T} \models R^- \sqsubseteq R''$,
- $(s, R', s'') \in \delta$ and $(s''', R'', s') \in \delta$, and
- $\exists R^- \in \mathsf{Loop}_\alpha[s'',s''']$ (after $k-1$ rule applications).

As $\mathsf{Tail}(d) = C$ and $\mathcal{T} \models C \sqsubseteq \exists R$, the element $d' = dR\exists R^-$ must belong to $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Then by applying the induction hypothesis, we can infer that there is a path p' that begins and ends at d', contains only elements from $\Delta^{\mathcal{I}_\mathcal{K}|_{d'}}$, and is such that $\lambda(p') \in L(\alpha_{s'',s'''})$. It follows that the path $p = dR'p'R''d$ satisfies all requirements, and in particular, is such that $\lambda(p) \in L(\alpha_{s,s'})$. (*end proof of claim*)

To show the other direction, we proceed by induction on the length of the path $p = e_0u_1e_1 \ldots u_ne_n$. The first base case is when $n = 0$, i.e., when $\lambda(p) = \varepsilon$. Then $\varepsilon \in L(\alpha_{s,s'})$, which implies that $s = s'$. By rule L1 of the definition of Loop_α, we must have $\mathsf{Tail}(d) \in \mathsf{Loop}_\alpha[s,s']$. The second base case is when $n = 1$, i.e., $p = dA?d$. Since p is a path, we must have $d \in A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, which means that $\mathcal{T} \models \mathsf{Tail}(d) \sqsubseteq A$. We also know that $\lambda(p) = A? \in L(\alpha_{s,s'})$, which implies that $(s, A?, s') \in \delta$. We have thus shown that the conditions of rule L3 are satisfied, and so $\mathsf{Tail}(d) \in \mathsf{Loop}_\alpha[s,s']$.

For the induction step, suppose that the second direction holds for all $0 \leq \ell < k$, and suppose that there is a path $p = e_0u_1e_1 \ldots u_ke_k$ in $\mathcal{I}_\mathcal{K}$ with $k > 2$ such that $\lambda(p) \in L(\alpha_{s,s'})$, $e_0 = e_k = d$, and $e_i \in \Delta^{\mathcal{I}_\mathcal{K}|_d}$ for every $0 \leq i \leq k$. First suppose that there exists some e_j with $0 < j < k$ such that $e_j = d$. Let $p_1 = e_0u_1e_1 \ldots e_j$ and $p_2 = e_ju_j \ldots e_k$. We know that $\lambda(p) = \lambda(p_1)\lambda(p_2) \in L(\alpha_{s,s'})$, so there must exist some state s'' such that $\lambda(p_1) \in L(\alpha_{s,s''})$ and $\lambda(p_2) \in L(\alpha_{s'',s'})$. Applying the induction hypothesis to p_1 and p_2, we obtain $\mathsf{Tail}(d) \in \mathsf{Loop}_\alpha[s'',s']$ and $\mathsf{Tail}(d) \in \mathsf{Loop}_\alpha[s,s'']$. Hence, by rule L2 of the construction of Loop_α, we must have $\mathsf{Tail}(d) \in \mathsf{Loop}_\alpha[s,s']$.

Now let us consider the second possibility, which is that $e_j \neq d$ for all $0 < j < k$. Since $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_d$ is tree-shaped, and p is a path, it must be the case that $e_1 = e_{k-1} = dRC \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}|_d}$. At this point, the proof slightly differs depending on whether we are in DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$. We present the proof for the case of $\mathcal{ELH}$, in which case we have $R = r \in \mathsf{N}_\mathsf{R}$ and $\mathcal{T} \models \mathsf{Tail}(d) \sqsubseteq \exists r.C$. As $\lambda(p) = u_1 \ldots u_k \in L(\alpha_{s,s'})$, $e_0 = e_k = d$, and $e_1 = e_{k-1} = dRC$, it must be the case that

- $u_1 \in \mathsf{N}_\mathsf{R}^\pm$ and $\mathcal{T} \models r \sqsubseteq u_1$, and

- $u_k \in \mathsf{N}_\mathsf{R}^\pm$ and $\mathcal{T} \models r \sqsubseteq t$ where $u_k = t^-$.

We also know that there must exist states $s'', s''' \in S$ such that

- $(s, u_1, s'') \in \delta$ and $(s''', u_k, s') \in \delta$, and
- $u_2 \dots u_{k-1} \in L(\alpha_{s'',s'''})$.

We can apply the induction hypothesis to $p' = e_1 u_1 \dots u_{k-1} e_{k-1}$ to infer that $\mathsf{Tail}(e_1) = C \in \mathsf{Loop}_\alpha[s'', s''']$. We thus satisfy all of the required conditions for applying rule L4' to obtain $\mathsf{Tail}(d) \in \mathsf{Loop}_\alpha[s, s']$. The proof for DL-Lite$_\mathcal{R}$ is analogous. □

Proposition 5.6. *For every DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, NFA α containing states s, s', and $a \in \mathsf{Ind}(\mathcal{A})$, the following statements are equivalent:*

1. *There is a concept $C \in \mathsf{ALoop}_\alpha[s, s']$ such that $\mathcal{T}, \mathcal{A} \models C(a)$.*
2. *There is a path $p = e_0 u_1 e_1 u_2 \dots u_n e_n$ in $\mathcal{I}_\mathcal{K}$ such that $\lambda(p) \in L(\alpha_{s,s'})$, $e_0 = e_n = a$, $n > 1$, and $e_i \notin \mathsf{Ind}(\mathcal{A})$ for $0 < i < n$.*

Proof. Fix a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$ and an NFA $\alpha = (S, \Sigma, \delta, s, F)$ (the proof for $\mathcal{ELH}$ is similar and left to reader). For the first direction, suppose that there is a concept $C \in \mathsf{ALoop}_\alpha[s, s']$ such that $\mathcal{T}, \mathcal{A} \models C(a)$. It follows from the definition of ALoop_α that there exists roles $R, R', R'' \in \mathsf{N}_\mathsf{R}^\pm$ and states $s'', s''' \in S$ such that $\mathcal{T} \models C \sqsubseteq \exists R$, $\mathcal{T} \models R \sqsubseteq R'$, $\mathcal{T} \models R^- \sqsubseteq R''$, $(s, R', s'') \in \delta$, $\exists R^- \in \mathsf{Loop}_\alpha[s'', s''']$, and $(s''', R'', s') \in \delta$. Since $\mathcal{T}, \mathcal{A} \models C(a)$ and $\mathcal{T} \models C \sqsubseteq \exists R$, it follows from the definition of $\mathcal{I}_\mathcal{K}$ that $aR\exists R^- \in \Delta^{\mathcal{I}_\mathcal{K}}$. By Proposition 5.5, $\exists R^- \in \mathsf{Loop}_\alpha[s'', s''']$ implies that there is a path $p = e_0 u_0 \dots u_n e_n$ in $\mathcal{I}_\mathcal{K}|_{aR\exists R^-}$ with $e_0 = e_n = aR\exists R^-$ and $\lambda(p) \in L(\alpha_{s'',s'''})$. Then it can be verified that by taking $p' = aR'pR''a$, we obtain a path in $\mathcal{I}_\mathcal{K}$ that satisfies the conditions of the second statement. In particular, since $(s, R', s'') \in \delta$, $\lambda(p) \in L(\alpha_{s'',s'''})$, and $(s''', R'', s') \in \delta$, we have $\lambda(p') \in L(\alpha_{s,s'})$.

For the second direction, suppose that there is a path $p = e_0 u_1 e_1 u_2 \dots u_n e_n$ in $\mathcal{I}_\mathcal{K}$ such that $\lambda(p) \in L(\alpha_{s,s'})$, $e_0 = e_n = a$, $n > 1$, and $e_i \notin \mathsf{Ind}(\mathcal{A})$ for $0 < i < n$. Since $n > 1$, $e_1 \notin \mathsf{Ind}(\mathcal{A})$, and p is a path, it must be the case that $e_1 = aR\exists R^-$ for some $R \in \mathsf{N}_\mathsf{R}^\pm$. Moreover, since $\mathcal{I}_\mathcal{K}|_a$ is a tree, we must have $e_{n-1} = e_1$. Then by the definition of paths, it must be the case that

- $u_1 \in \mathsf{N}_\mathsf{R}^\pm$ and $\mathcal{T} \models R \sqsubseteq u_1$, and
- $u_n \in \mathsf{N}_\mathsf{R}^\pm$ and $\mathcal{T} \models R^- \sqsubseteq u_n$.

We also know that there must exist states $s'', s''' \in S$ such that:

- $(s, u_1, s'') \in \delta$ and $(s''', u_n, s') \in \delta$
- $u_2 \dots u_{n-1} \in L(\alpha_{s'',s'''})$

By applying Proposition 5.5 to $p' = e_1 u_1 \dots u_{n-1} e_{n-1}$, we can infer that $\exists R^- \in \mathsf{Loop}_\alpha[s'', s''']$. Since $aR\exists R^- \in \Delta^{\mathcal{I}_\mathcal{K}}$, the definition of $\mathcal{I}_\mathcal{K}$ ensures that there is some $C \in \mathsf{BC}_\mathcal{T}$ such that $a \in C^{\mathcal{I}_\mathcal{K}}$ and $\mathcal{T} \models C \sqsubseteq \exists R$. The conditions of Rule 5 are thus satisfied, so $C \in \mathsf{ALoop}_\alpha[s, s']$. □

```
ALGORITHM EvalAtom
INPUT: NFA $\alpha = (S, \Sigma, \delta, s_0, F)$ with $\Sigma \subseteq \mathsf{N}_\mathsf{R}^\pm \cup \{A? \mid A \in \mathsf{N_C}\}$, DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB
       $(\mathcal{T}, \mathcal{A})$, $(a, b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A})$

1. Test whether $(\mathcal{T}, \mathcal{A})$ is satisfiable, output yes if not.

2. Initialize $\mathsf{current} = (a, s_0)$ and $\mathsf{count} = 0$. Set $\mathsf{max} = |\mathcal{A}| \cdot |S| + 1$.

3. While $\mathsf{count} < \mathsf{max}$ and $\mathsf{current} \notin \{(b, s_f) \mid s_f \in F\}$

   (a) Let $\mathsf{current} = (c, s)$.
   (b) Guess a pair $(d, s') \in \mathsf{Ind}(\mathcal{A}) \times S$ together with either $(s, \sigma, s') \in \delta$ or $B \in$
       $\mathsf{ALoop}_\alpha[s, s']$.
   (c) If $(s, \sigma, s')$ was guessed
       • If $\sigma \in \mathsf{N}_\mathsf{R}^\pm$, then verify that $\mathcal{T}, \mathcal{A} \models \sigma(c, d)$, and return no if not.
       • If $\sigma = A?$, then verify that $c = d$ and $\mathcal{T}, \mathcal{A} \models A(c)$, and return no if not.
   (d) If $B$ was guessed, then verify that $c = d$ and $\mathcal{T}, \mathcal{A} \models B(c)$, and return no if not.
   (e) Set $\mathsf{current} = (d, s')$ and increment $\mathsf{count}$.

4. If $\mathsf{current} = (b, s_f)$ for some $s_f \in F$, return yes. Else return no.
```

Figure 11: Non-deterministic algorithm for 2RPQ answering in DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$.

Now that we have a means to determine which loops through the anonymous part are available from a given ABox individual, we are ready to present the extended evaluation algorithm EvalAtom in Figure 11 that handles DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ KBs. The algorithm EvalAtom differs from BasicEval in two respects. First, because DL-Lite$_\mathcal{R}$ KBs may contain contradictions, there is an initial consistency check in Step 1 to determine whether the input KB is satisfiable (this step can be skipped for $\mathcal{ELH}$ KBs, which are always satisfiable). If the KB is shown to be unsatisfiable, then the query trivially holds, so the algorithm outputs yes. The second difference occurs in Step 3(b) within the while loop, where we now have the choice between guessing a pair $(d, s') \in \mathsf{Ind}(\mathcal{A}) \times S$ (as before) and guessing a concept $B \in \mathsf{ALoop}_\alpha[s, s']$. The first option corresponds to taking a step in the ABox, whereas the second corresponds to a shortcut through the anonymous part. If we choose the second option, then we must check that the selected concept is entailed at the current individual. The exit conditions for the while loop and the criterion for outputting yes in Step 4 remain unchanged.

Example 5.7. Algorithm EvalAtom correctly returns yes on the input (a, a) together with our example query and KB, in contrast to BasicEval (as was shown in Example 5.3). Indeed, a successful execution (illustrated pictorially in Figure 12) starts from (a, s_0) and guesses the pair (b, s_0) and the transition (s_0, r, s_0) in the first iteration of the while loop. The checks in Step 3(c) succeed as $\mathcal{T}, \mathcal{A} \models r(a, b)$. In the next iteration, it can guess (b, s_2) and, as $\exists r^- \in \mathsf{ALoop}_\alpha[s_0, s_2]$, it can also guess the concept $\exists r^-$. The checks in Step 3(d) succeed because $\mathcal{T}, \mathcal{A} \models \exists r^-(b)$. In the third iteration, it guesses the pair (a, s_f) and the transition

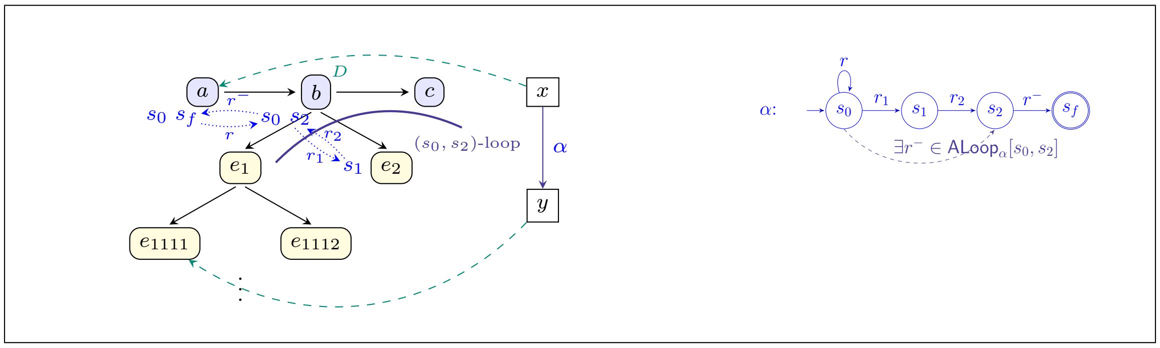

Figure 12: Establishing $(a, a) \in \mathsf{cert}(q, \mathcal{K})$ using $\mathsf{EvalAtom}$ for $q(x, y) = \alpha(x, y)$

(s_2, r^-, s_f). As $\mathcal{T}, \mathcal{A} \models r^-(b, a)$, the checks in Step 3(c) succeed again. Since $s_f \in F$, this is the last iteration, and in Step 4 the algorithm returns `yes`.

Proposition 5.8. *For every 2RPQ $q = \alpha(x, y)$, DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and pair of individuals (a, b) from $\mathsf{Ind}(\mathcal{A})$: $(a, b) \in \mathsf{cert}(q, \mathcal{K})$ if and only if there is some execution of $\mathsf{EvalAtom}(\alpha, \mathcal{K}, (a, b))$ that returns* `yes`.

Proof. Consider a 2RPQ $q = \alpha(x, y)$ with $\alpha = (S, \Sigma, \delta, s_0, F)$, a DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and a pair of individuals (a, b) from $\mathsf{Ind}(\mathcal{A})$.

For the first direction, suppose that $(a, b) \in \mathsf{cert}(q, \mathcal{K})$. Then there is a path $p = e_0 u_1 \dots u_n e_n$ in $\mathcal{I}_\mathcal{K}$ such that $e_0 = a$, $e_n = b$, and $\lambda(p) \in L(\alpha)$. We may assume without loss of generality that there is no shorter path with these properties. Since $\lambda(p) \in L(\alpha)$, we can find a sequence of states $s_0 s_1 \dots s_n$ from S such that $s_n \in F$ and for every $1 \leq i \leq n$, $(s_{i-1}, u_i, s_i) \in \delta$. Let $j_1 < \dots < j_m$ be all of the indices i such that $e_i \in \mathsf{Ind}(\mathcal{A})$, and consider the sequence of pairs $\Gamma = (e_{j_1}, s_{j_1})(e_{j_2}, s_{j_2}) \dots (e_{j_m}, s_{j_m})$. Observe that $j_1 = 0$ and $j_m = n$, so we have $e_{j_1} = a$, $e_{j_m} = b$, and $s_{j_m} \in F$. Also observe that we must have $(e_{j_\ell}, s_{j_\ell}) \neq (e_{j_k}, s_{j_k})$ whenever $\ell \neq k$, since otherwise, we could construct a shorter path with the same properties as p, contradicting our minimality assumption. It follows that the sequence Γ contains at most $|\mathcal{A}| \cdot |S|$ pairs. Thus, to prove that the sequence Γ leads to an execution of $\mathsf{EvalAtom}$ that returns `yes`, it only remains to show that it is always possible to guess a transition or concept in 3(b) such that the checks in 3(c) and 3(d) succeed. Thus, let us suppose that $\mathsf{current} = (e_{j_\ell}, s_{j_\ell})$ and we guess the pair $(e_{j_{\ell+1}}, s_{j_{\ell+1}})$ in 3(b). There are three cases to consider:

- Case 1: $j_{\ell+1} = j_\ell{+}1$ and $u_{j_{\ell+1}} = R \in \mathsf{N}_\mathsf{R}^\pm$. From earlier, we have that $(s_{j_\ell}, R, s_{j_{\ell+1}}) \in \delta$, and since p is a path, we know that $(e_{j_\ell}, e_{j_{\ell+1}}) \in R^{\mathcal{I}_\mathcal{K}}$. The latter implies that $\mathcal{T}, \mathcal{A} \models R(e_{j_\ell}, e_{j_{\ell+1}})$, so by choosing the transition $(s_{j_\ell}, R, s_{j_{\ell+1}})$ in 3(b), we can ensure that the entailment check will succeed in 3(c).
- Case 2: $j_{\ell+1} = j_\ell + 1$ and $u_{j_{\ell+1}} = A?$. From earlier, we have that $(s_{j_\ell}, A?, s_{j_{\ell+1}}) \in \delta$, and since p is a path, we know that $e_{j_\ell} = e_{j_{\ell+1}} \in A^{\mathcal{I}_\mathcal{K}}$. By choosing the transition $(s_{j_\ell}, A?, s_{j_{\ell+1}})$, we can ensure that the conditions of 3(c) are satisfied.
- Case 3: $j_{\ell+1} > j_\ell + 1$. In this case, the path $p' = e_{j_\ell} u_{j_\ell+1} \dots u_{j_{\ell+1}} e_{j_{\ell+1}}$ is such that $\lambda(p') \in L(\alpha_{s_{j_\ell}, s_{j_{\ell+1}}})$, $e_{j_\ell} = e_{j_{\ell+1}} \in \mathsf{Ind}(\mathcal{A})$, and for every $j_\ell < i < j_{\ell+1}$, we have

$e_i \notin \mathsf{Ind}(\mathcal{A})$. We can thus apply Lemma 5.6 to find a concept $C \in \mathsf{ALoop}_\alpha[s_{j_\ell}, s_{j_{\ell+1}}]$ such that $\mathcal{T}, \mathcal{A} \models C(e_{j_\ell})$. By choosing C in 3(b), we can be sure that the entailment check in 3(d) will succeed.

For the other direction, consider some execution of $\mathsf{EvalAtom}(\alpha, \mathcal{K}, (a, b))$ that returns `yes`, and let $(c_0, s_0)(a_1, s_1) \dots (c_n, s_n)$ be the sequence of pairs that were guessed by this execution. Then we must have $c_0 = a$, $c_n = b$, and $s_n \in F$. To complete the proof, it suffices to establish the following claim:

Claim: For every $0 \leq i \leq n$, there is a path p_i from c_0 to c_i in $\mathcal{I}_\mathcal{K}$ such that $\lambda(p_i) \in L(\alpha_{s_0,s_i})$.

Proof of claim. The proof is by induction on i. For the base case ($i = 0$), we can simply take the path $p_0 = c_0$ since $\lambda(p_0) = \varepsilon$ and $\varepsilon \in L(\alpha_{s_0,s_0})$. For the induction step, we suppose that p_{k-1} is a path from c_0 to c_{k-1} such that $\lambda(p_{k-1}) \in L(\alpha_{s_0,s_{k-1}})$, and we show how to construct a path p_k with the required properties. There are three cases to consider depending on which transition or concept was guessed together with (c_k, s_k) in Step 3(b).

- Case 1: the transition $(s_{k-1}, R, s_k) \in \delta$ was guessed. Since the check in 3(c) succeeded, we have $\mathcal{T}, \mathcal{A} \models R(c_{k_1}, c_k)$. Then $p_k = p_{k-1}Rc_k$ is a path in $\mathcal{I}_\mathcal{K}$ from c_0 to c_k such that $\lambda(p_k) \in L(\alpha_{s_0,s_k})$.
- Case 2: the transition $(s_{k-1}, A?, s_k) \in \delta$ was guessed. Then since the check in 3(c) succeeded, we have $c_k = c_{k-1}$ and $\mathcal{T}, \mathcal{A} \models A(c_{k_1})$. Then $p_k = p_{k-1}A?c_k$ is a path in $\mathcal{I}_\mathcal{K}$ from c_0 to c_k such that $\lambda(p_k) \in L(\alpha_{s_0,s_k})$.
- Case 3: the concept $B \in \mathsf{ALoop}_\alpha[s_{k-1}, s_k]$ was guessed. Since the check in 3(d) succeeded, we have $c_k = c_{k-1}$ and $\mathcal{T}, \mathcal{A} \models B(c_{k-1})$. By applying Lemma 5.6, we can find a path $p' = e_0u_1e_1u_2 \dots u_ne_n$ in $\mathcal{I}_\mathcal{K}$ such that $\lambda(p') \in L(\alpha_{s_{k-1},s_k})$ and $e_0 = e_n = c_{k-1}$. Then $p_k = p_{k-1}u_1e_1u_2 \dots u_ne_n$ is a path from c_0 to c_k with $\lambda(p_k) = \lambda(p_{k-1})\lambda(p') \in L(\alpha_{s_0,s_k})$. □

By analyzing the complexity of our modified evaluation procedure, we obtain upper bounds that match the lower bounds from Section 4.

Theorem 5.9. *2RPQ answering is in*

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

Proof. The procedure $\mathsf{EvalAtom}$ involves three different types of checks: the consistency check performed in Step 1 and the entailment and loop checks that take place in Step 3. The cost of these checks depends on the choice of the DL and the complexity measure. Aside from these checks, the base procedure runs in non-deterministic logarithmic space in combined complexity, as only logarithmic space is needed to keep track of the value of `count`, the old and new values of `current`, and the guesses of transitions and concepts.

For DL-Lite$_\mathcal{R}$, we know from existing results (Calvanese et al., 2007) that the consistency and entailment checks can be performed in non-deterministic logarithmic space in combined complexity (and hence also for data complexity). We have also seen that the table ALoop_α can be constructed in polynomial time in $|\mathcal{T}|$ and α, so the loop checks can be performed in constant time in $|\mathcal{A}|$ and in polynomial time w.r.t. the whole input. It follows that $\mathsf{EvalAtom}$ runs in non-deterministic logarithmic space w.r.t. $|\mathcal{A}|$, yielding Statement (1). Regarding Statement (2), we note that $\mathsf{EvalAtom}$ can be viewed as an NL procedure that uses a P oracle to handle the loop checks. Since $\mathrm{NL}^\mathrm{P} = \mathrm{P}$, this yields a P upper bound on the combined complexity of 2RPQ answering in DL-Lite$_\mathcal{R}$. The proof of Statement (3) is similar. We simply note that when the input TBox is formulated in $\mathcal{ELH}$, all three types of checks can be performed in polynomial time w.r.t. the whole input (Baader et al., 2005). Using $\mathrm{NL}^\mathrm{P} = \mathrm{P}$, we may conclude that $\mathsf{EvalAtom}$ provides a polynomial-time procedure for 2RPQ answering in $\mathcal{ELH}$. $\square$

6. Upper Bounds for C2RPQs

The main objective of this section is to define a procedure for deciding, given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, C2RPQ $q(\vec{x})$ of arity k, and k-tuple $\vec{a}$ of individuals from $\mathcal{A}$, whether there is a match π for q in $\mathcal{I}_\mathcal{K}$ such that $\pi(\vec{x}) = \vec{a}$. A naïve approach might consist in guessing a mapping π from the query variables to the individuals in the core of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ and then checking that π is a match by running the $\mathsf{EvalAtom}$ algorithm on $\alpha(\pi(t), \pi(t'))$ for every query atom $\alpha(t, t')$. Such an algorithm would properly take into account paths between individuals that pass by the anonymous part, but because it does not consider matches that send variables to anonymous objects, it would still be incomplete. In particular, such a procedure would not return a or b as answers in Example 3.4. To regain completeness, one could instead guess matches into the entire domain of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, but this would not yield a decision procedure since the latter may be infinite. Moreover, since matches of C2RPQs can involve domain elements that are arbitrarily far apart, it is not apparent how to identify a suitable finite subset of the domain that is guaranteed to contain a match for the query if one exists. To address these challenges, the procedure we propose in this section comprises two main steps. As a first step, we rewrite the input query q into a set Q of C2RPQs such that there is a match π for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ with $\pi(\vec{x}) = \vec{a}$ if and only if there is a match π' for some $q' \in Q$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ with $\pi'(\vec{x}) = \vec{a}$. The advantage of the rewritten queries is that we will only need to consider matches π' which map query variables to $\mathsf{Ind}(\mathcal{A})$. The second step decides the existence of such restricted matches for the rewritten queries using the $\mathsf{EvalAtom}$ procedure defined in Section 5.

For the purposes of this section, it will prove convenient to work with DL-Lite$_\mathcal{R}$ TBoxes that satisfy the following condition: for every role name $r \in \mathsf{sig}(\mathcal{T})$, there exists concept names A_r, A_r^- such that $\mathcal{T}$ contains the inclusions $A_r \sqsubseteq \exists r$, $\exists r \sqsubseteq A_r$, $A_{r^-} \sqsubseteq \exists r^-$, $\exists r^- \sqsubseteq A_{r^-}$, and these are the only inclusions in $\mathcal{T}$ involving the concept names A_r and A_{r^-}. Note that if $\mathcal{T}$ does not satisfy this condition, then we can simply choose fresh concepts A_r, A_{r^-} for each role name $r \in \mathsf{sig}(\mathcal{T})$ and add the corresponding inclusions to $\mathcal{T}$. The resulting TBox $\mathcal{T}'$ is a model conservative of extension of $\mathcal{T}$, hence for every ABox $\mathcal{A}$ and every C2RPQ q with $\mathsf{sig}(q) \cap (\mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})) = \emptyset$, we have $\mathsf{cert}(q, (\mathcal{T}, \mathcal{A})) = \mathsf{cert}(q, (\mathcal{T}', \mathcal{A}))$. We may therefore assume without any loss of generality that all DL-Lite$_\mathcal{R}$ TBoxes considered in this section satisfy this syntactic condition. In what follows, when we use a concept name A_R, with $R \in \mathsf{N}_\mathsf{R}^\pm$, we will assume the TBox contains the inclusions $A_R \sqsubseteq \exists R$ and $\exists R \sqsubseteq A_R$.

6.1 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-$\mathcal{SHIQ}$ introduced by Eiter, Ortiz, Šimkus, Tran, and Xiao (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 in such a way that the resulting query q' has a match π' that is the same as π except that those variables mapped to $\pi(y)$ by π are now mapped by π' to the (unique) parent of $\pi(y)$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. The delicate point is that we must "split" atoms of the form $\alpha(t, t')$ with $y \in \{t, t'\}$ into the parts which are satisfied in the subtree $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_{\pi(y)}$, 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 obtain a query that has a match which maps all terms to $\mathsf{Ind}(\mathcal{A})$.

In Figure 13, we implement this intuition by defining an algorithm OneStep that performs a single (non-deterministic) rewriting step. We illustrate the functioning of OneStep in the following examples.

Example 6.1. Recall the query $q(x) = \exists y, z.\ r^* \cdot r_1 \cdot r_2 \cdot r^-(x, y) \wedge r^- \cdot r^-(y, z) \wedge D(z)$ from Example 3.4, and the KB $(\mathcal{T}, \mathcal{A})$ from Example 2.4.

To illustrate the rewriting algorithm, we first disregard the first atom and consider the simpler Boolean query $q_1 = \exists y, z.\ \beta(y, z) \wedge D(z)$, where β is the NFA with language $r^- \cdot r^-$ depicted in Figure 14. The figure shows a match π for q_1 with $\pi(y) = e_{11}$ and $\pi(z) = b$. Since y is a leaf of the image of q_1 under π, we want to modify q_1 into a query q_1' that has a match π' which only differs from π in having $\pi'(y) = e_1$, where e_1 is the parent of $\pi(y) = e_{11}$. Intuitively, this is done using OneStep by choosing $\mathsf{Leaf} = \{y\}$ as the only variable to be 'moved up'. Then we choose the concept B because if it holds at e_1, then this enforces the existence of an r^- role from a child of e_1 (namely, e_{11}) to e_1. Hence, by checking that B holds at e_1, we implicitly check that the first r^- needed to satisfy β holds below e_1. This rewriting step is illustrated in the upper part of Figure 14. Formally, in Step 1 we choose $\mathsf{Leaf} = \{y\}$. Then in Step 2 we choose $\exists r^-$ (since, intuitively, this is the tail concept that 'causes' everything that holds at e_{11}). In Step 3, we simply take the final state s_f', so the atom $\beta(z, y)$ remains the same, and in Step 4 we do nothing. These two steps are so simple in this example because there are no complex paths in our query match that need to be deeper than e_{11}. Next, in Step 5, we choose B, since $\mathcal{T} \models \exists r$, that is, B enforces the existence of a node that satisfies the tail concept $\exists r^-$ that we guessed above. In Step 5b we only need to take care of the atom $\beta(y, z)$. We choose s_1' and in Step 6, we replace β by $\beta_{s_1', s_f'}$. In Step 7, we add the atom $B(y)$, and we output the query $B(y) \wedge \beta_{s_1', s_f'}(y, z) \wedge D(z)$.

The lower part of the figure illustrates a successive application of OneStep in which we proceed analogously, dropping the second r^- of β, and adding again an atom $B(y)$. This results in the query $q_1'' = \exists y, z.\ B(y)\beta_{s_f', s_f'}(y, z) \wedge D(z)$, which has a match $\pi''(z) = \pi''(y) = b$ ranging over individuals only. Note that $L(\beta_{s_f', s_f'}) = \{\varepsilon\}$ so this query is equivalent to $\exists y.B(y) \wedge D(y)$.

```
ALGORITHM OneStep
INPUT: C2RPQ $q$ with all binary atoms specified by NFAs, DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ TBox $\mathcal{T}$

1. Guess a non-empty set $\mathsf{Leaf} \subseteq \mathsf{qvars}(q)$ and $y \in \mathsf{Leaf}$.
   Rename all variables in $\mathsf{Leaf}$ to $y$.

2. Guess $C \in \mathsf{TC}_\mathcal{T}$ such that $\mathcal{T} \models C \sqsubseteq B$ for every atom $B(y) \in q$. Drop all such atoms
   from $q$.

3. For each atom $\alpha(t,t')$ where $\alpha = (S,\Sigma,\delta,s,F)$ is an NFA and $y \in \{t,t'\}$,

   • Guess a sequence $s_1,\dots,s_{n-1}$ of distinct states from $S$ and a state $s_n \in F$.
   • Replace $\alpha(t,t')$ by the atoms $\alpha_{s,s_1}(t,y)$, $\alpha_{s_1,s_2}(y,y)$, ..., $\alpha_{s_{n-2},s_{n-1}}(y,y)$,
     $\alpha_{s_{n-1},s_n}(y,t')$.

4. Drop all atoms $\alpha_{s,s'}(y,y)$ such that $C \in \mathsf{Loop}_\alpha[s,s']$.

5. Guess some $D \in \mathsf{BC}_\mathcal{T}$ and $R \in \mathsf{N}_\mathsf{R}^\pm$ such that:

   (a) $C = \exists R^-$ and $\mathcal{T} \models D \sqsubseteq \exists R$    (if $\mathcal{T}$ is a DL-Lite$_\mathcal{R}$ TBox)
       $R \in \mathsf{N}_\mathsf{R}$ and $\mathcal{T} \models D \sqsubseteq \exists R.C$    (if $\mathcal{T}$ is an $\mathcal{ELH}$ TBox)
   (b) For each atom $\alpha(y,t)$ with $\alpha = (S,\Sigma,\delta,s,F)$, there exists $s' \in S$ and $U \in \mathsf{N}_\mathsf{R}^\pm$
       with $(s,U^-,s') \in \delta$ and $\mathcal{T} \models R \sqsubseteq U$.
   (c) For each atom $\alpha(t,y)$ with $\alpha = (S,\Sigma,\delta,s,F)$, there exists $s'' \in S$, $s_f \in F$, and
       $U \in \mathsf{N}_\mathsf{R}^\pm$ with $(s'',U,s_f) \in \delta$ and $\mathcal{T} \models R \sqsubseteq U$.

   For atoms of the form $\alpha(y,y)$, conditions (b) and (c) must both be satisfied.

6. Replace

   • each atom $\alpha(y,t)$ with $t \neq y$ by $\alpha_{s',F}(y,t)$
   • each atom $\alpha(t,y)$ with $t \neq y$ by $\alpha_{s,s''}(t,y)$
   • each atom $\alpha(y,y)$ by $\alpha_{s',s''}(y,y)$

   with $s,s',s'',F$ as in Step 5.

7. If $D \in \mathsf{N}_\mathsf{C}$ is the concept chosen in Step 5, add $D(y)$ to $q$. If $D = \exists P^-$ was the chosen
   concept, add $A_{P^-}(y)$ to $q$. Output $q$.
```

Figure 13: Non-deterministic query rewriting algorithm OneStep.

Example 6.2. Now we illustrate two rewriting steps for $q(x) = \exists y,z.\ r^* \cdot r_1 \cdot r_2 \cdot r^-(x,y) \wedge r^- \cdot r^-(y,z) \wedge D(z)$ (see Figure 15). First, recall the match π from Figure 7, which is also reproduced in the top part of Figure 15. Observe that $\pi(y) = e_{11}$ is a leaf in the image of π. To move the match up one step, we make the same initial choices as in Example 6.1, setting $\mathsf{Leaf} = \{y\}$ in Step 1 and selecting the tail concept $\exists r^-$ that characterises e_{11} in

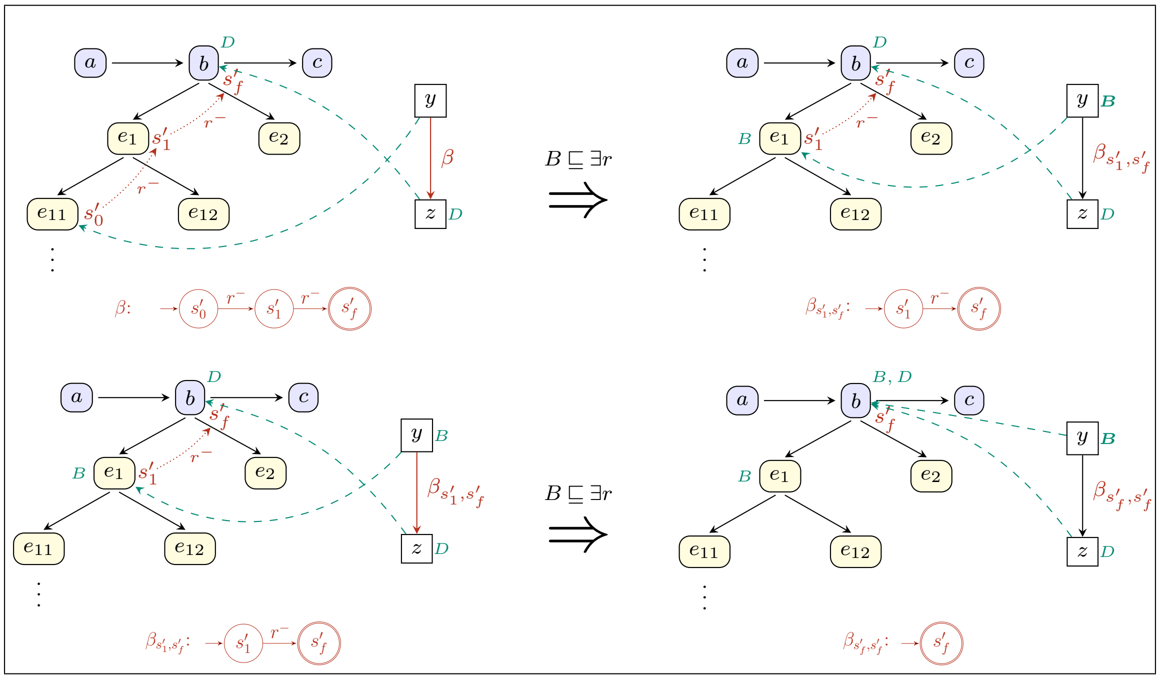

Figure 14: Two successive applications of OneStep to $q_1 = \exists y, z. r^- \cdot r^-(y, z) \wedge D(z)$

Step 2. However, because the path that satisfies α goes deeper than $\pi(y) = e_{11}$, in Step 3, we will need to separate the path into the parts that occur below e_{11} and those that occur above it. We choose the sequence of states s_0, s_f as these correspond to the states of the automata when it visits e_{11}, and we replace $\alpha(x, y)$ by $\alpha_{s_0,s_0}(x, y)$ and $\alpha_{s_0,s_f}(y, y)$. Note that this intermediate query is illustrated at the top of Figure 15. Since $\exists r^- \in \mathsf{Loop}_\alpha[s_0, s_f]$ (see Example 5.4), we drop the second atom in Step 4. Now we can 'move up' y similarly as above, but considering simultaneously atoms $\alpha_{s_0,s_0}(x, y)$ and $\beta(y, z)$. In Step 5, we again choose the concept B, since it satisfies $\mathcal{T} \models B \sqsubseteq \exists r$. For the atom $\beta(y, z)$ we choose s'_1 and $U = r$ (Step 5(b)) and for $\alpha_{s_0,s_0}(x, y)$, we choose s_0 and $U = r$ (Step 5(c)). In Step 6, we replace the two query atoms by $\alpha_{s_0,s_0}(x, y)$ and $\beta_{s'_1,s'_f}(y, z)$, and in Step 7, we add the atom $B(y)$ and output the resulting query $\alpha_{s_0,s_0}(x, y) \wedge B(y) \wedge \beta_{s'_1,s'_f}(y, z)$, which is displayed in the middle of Figure 15. The lower part of this figure depicts a second, similar application of OneStep that outputs a query that has a match ranging over the individuals only.

Slightly abusing notation, we will use $\mathsf{OneStep}(q, \mathcal{T})$ to denote the set of queries that are output by some execution of OneStep on input $(q,\mathcal{T})$. We then consider the set $\mathsf{Rewrite}(q, \mathcal{T})$ consisting of all queries that can be obtained from $(q,\mathcal{T})$ by zero or more applications of OneStep. Formally, we define $\mathsf{Rewrite}(q, \mathcal{T})$ as the smallest set that contains the initial query q and is closed under applications of OneStep, i.e., if $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ and $q'' \in \mathsf{OneStep}(q', \mathcal{T})$, then $q'' \in \mathsf{Rewrite}(q, \mathcal{T})$.

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

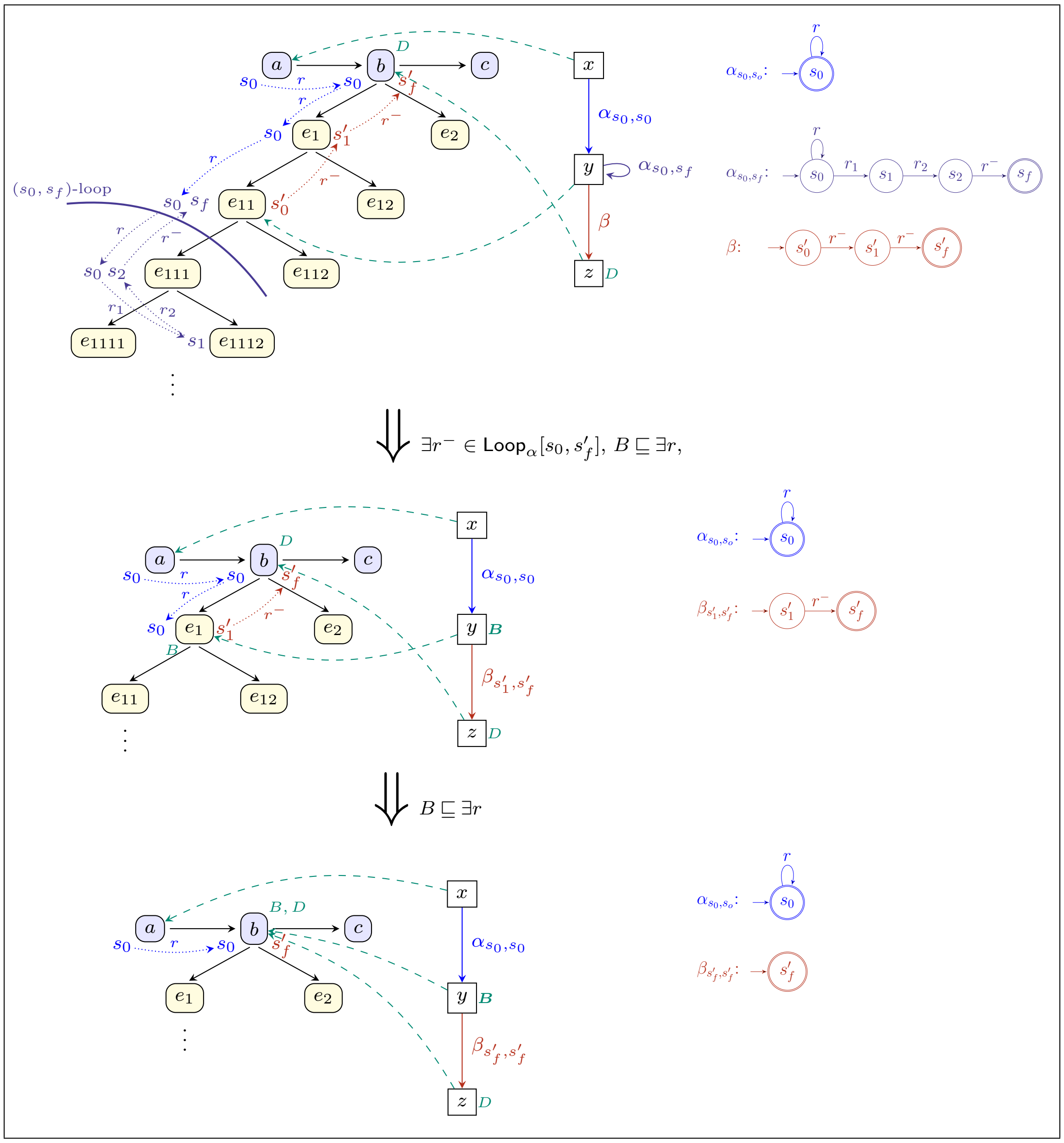
(s_0, s_f)-loop
$\exists r^- \in \mathsf{Loop}_\alpha[s_0, s'_f]$, $B \sqsubseteq \exists r$,
$B \sqsubseteq \exists r$
α_{s_0,s_0}
α_{s_0,s_f}
β
$\beta_{s'_1,s'_f}$
$\beta_{s'_f,s'_f}$

Figure 15: Two rewriting steps for $q(x) = \exists y, z.\ r^* \cdot r_1 \cdot r_2 \cdot r^-(x, y) \wedge r^- \cdot r^-(y, z) \wedge D(z)$

Proposition 6.3. *For every satisfiable DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $(\mathcal{T}, \mathcal{A})$ and C2RPQ q: $\vec{a} \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$ if and only if there exists a query $q'(\vec{x}) \in \mathsf{Rewrite}(q, \mathcal{T})$ and a match π for q' in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi(\vec{x}) = \vec{a}$ and $\pi(z) \in \mathsf{Ind}(\mathcal{A})$ for every variable z in q'.*

We split Proposition 6.3 into two lemmas, the first showing completeness of Rewrite, and the second showing its correctness.

Lemma 6.4. *If $\vec{a} \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$, then there exists a query $q'(\vec{x}) \in \mathsf{Rewrite}(q, \mathcal{T})$ and a match π for q' in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi(\vec{x}) = \vec{a}$ and $\pi(z) \in \mathsf{Ind}(\mathcal{A})$ for every variable z in q'.*

Proof. Consider a knowledge base $(\mathcal{T}, \mathcal{A})$ and its canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. For every element $e \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, we define the distance $\mathsf{dist}(e)$ of e from the core of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ as follows: $\mathsf{dist}(aR_1C_1 \dots R_nC_n) = n$. Observe that $\mathsf{dist}(e) = 0$ implies $e \in \mathsf{Ind}(\mathcal{A})$. Using this notion of distance, we can define the *cost of a match* μ of some query in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ as $\Sigma_{v \in \mathsf{dom}(\mu)}$ $\mathsf{dist}^{\mathcal{T},\mathcal{A}}(\mu(v))$, where $\mathsf{dom}(\mu)$ the domain of μ. We remark that a match has cost equal to zero just in the case that it maps all query variables to ABox individuals.

Suppose that $\vec{a} \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$, and π is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi(\vec{x}) = \vec{a}$. If π maps all variables in q to $\mathsf{Ind}(\mathcal{A})$, then we are done. Otherwise, there must exist some variable y such that $\pi(y) = eSC$ and there is no $z \in \mathsf{vars}(q)$ such that $\pi(y)$ is a proper prefix of $\pi(z)$. We aim to construct a match π' for some query $q' \in \mathsf{OneStep}(q, \mathcal{T})$ such that

(c1) $\pi'(t) = \pi(t)$ for every $t \in \mathsf{terms}(q)$ such that $\pi(t) \neq \pi(y)$; and

(c2) $\pi'(t) = e$ for every $t \in \mathsf{terms}(q)$ with $\pi(t) = \pi(y)$.

In other words, π' is essentially the same as π except that it maps each t with $\pi(t) = \pi(y)$ one step closer to the ABox. Observe that (c1) and (c2) together ensure that the cost of π' is strictly inferior to the cost of π. Thus, by repeatedly applying this operation, we will eventually obtain a query $q^* \in \mathsf{Rewrite}(q, \mathcal{T})$ and a match π^* for q^* that has cost zero, i.e., such that $\pi^*(z) \in \mathsf{Ind}(\mathcal{A})$ for every variable z in q^*.

We now show how to obtain a query $q' \in \mathsf{OneStep}(q, \mathcal{T})$ and match π' with properties (c1) and (c2). In Step 1, we set $\mathsf{Leaf} = \{t \in \mathsf{terms}(q) \mid \pi(t) = \pi(y) = eSC\}$. Note that $\mathsf{Leaf} \subseteq \mathsf{qvars}(q)$ since $eSC \notin \mathsf{Ind}(\mathcal{A})$. We define a function σ as follows: $\sigma(t) = t$ if $\pi(t) \neq \pi(y)$, else $\sigma(t) = y$. At the end of Step 1, we have the query:

$$\{B(\sigma(t)) \mid B(t) \in q_0\} \cup \{\alpha(\sigma(t), \sigma(t')) \mid \alpha(t, t') \in q_0\}$$

In Step 2, we choose the concept C (note that $C \in \mathsf{TC}_{\mathcal{T}}$ because $\pi(y) = eSC$). Consider some atom $B(y)$ that is present at the end of Step 1. The existence of atom $B(y)$ at the end of this step means that there must have existed an atom $B(v) \in q$ with $v \in \mathsf{Leaf}$. Since π is a match for q, we know that $\pi(v) = \pi(y) \in B^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. As $\pi(y) = eSC$, it follows from the definition of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ that $\mathcal{T} \models C \sqsubseteq B$, as required by Step 2.

Next we show how to select states $s_1, \dots, s_n$ in Step 3. Consider an atom $\alpha(t_1, t_2)$ which is present in the query at the start of Step 3, such that $y \in \{t_1, t_2\}$ and $\alpha = (S, \Sigma, \delta, s_0, F)$. Then we know that there is an atom $\alpha(t_1', t_2')$ in the input query q such that $t_1 = \sigma(t_1')$ and $t_2 = \sigma(t_2')$. Since π is a match for q with $\pi(t_1) = \pi(t_1')$ and $\pi(t_2) = \pi(t_2')$, we know that $(\pi(t_1), \pi(t_2)) \in L(\alpha)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. It follows that we can find a path $p = e_0u_1e_1 \dots u_me_m$ such that $e_0 = \pi(t_1)$, $e_m = \pi(t_2)$, and $\lambda(p) \in L(\alpha)$. We assume without loss of generality that m is minimal, i.e., we cannot find a path satisfying the same properties but of shorter length. Now let $j_1 < \dots < j_{n-1}$ be all of the indices $0 < \ell < m$ such that $e_\ell = \pi(y)$, and consider the following paths:

- $p_1 = e_0u_1 \dots u_{j_1}e_{j_1}$
- $p_i = e_{j_{i-1}}u_{j_{i-1}+1} \dots u_{j_i}e_{j_i}$ for $1 < i < n$
- $p_n = e_{j_{n-1}}u_{j_{n-1}+1} \dots u_me_m$

We also define a sequence of states $s_1, \ldots, s_n$ such that

- $\lambda(p_1) \in L(\alpha_{s_0,s_1})$,
- $\lambda(p_i) \in L(\alpha_{s_{i-1},s_i})$ for $1 < i \leq n$, and
- $s_n \in F$.

Note that such a sequence of states must exist since $\lambda(p) = \lambda(p_1)\lambda(p_2)\ldots\lambda(p_n) \in L(\alpha)$ and α has start state s_0 and final states F. Using the fact that $e_{j_i} = \pi(y)$ for $1 \leq i < n$, we have the following:

- $(\pi(t_1), \pi(y)) = (\pi(t_1'), \pi(y)) \in L(\alpha_{s_0,s_1})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.
- $(\pi(y), \pi(y)) \in L(\alpha_{s_{i-1},s_i})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, for $2 \leq i < n$. $(\star)$
- $(\pi(y), \pi(t_2)) = (\pi(y), \pi(t_2')) \in L(\alpha_{s_{n-1},s_n})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

We aim to show that $s_i \neq s_j$ for every $1 \leq k < \ell < n$. Suppose for a contradiction that $s_k = s_\ell$ for some $1 \leq k < \ell < n$, and consider the path $p' = e_0u_1 \ldots u_{j_k}e_{j_k}u_{j_\ell+1}e_{j_\ell+1} \ldots u_me_m$. As $\lambda(p') = u_1 \ldots u_{j_k}u_{j_\ell+1} \ldots u_m$, $u_1 \ldots u_{j_k} \in L(\alpha_{s_0,s_k})$, $u_{j_\ell+1} \ldots u_m \in L(\alpha_{s_\ell,s_n}) = L(\alpha_{s_k,s_n})$, and $s_n \in F$, it follows that $\lambda(p') \in L(\alpha)$. However, this means that p' satisfies the same conditions as p but has strictly shorter length, contradicting our minimality assumption. Hence all states in the sequence $s_1, \ldots, s_{n-1}$ must be distinct. We can thus choose this sequence of states in Step 3, and replace the atom $\alpha(t_1, t_2) = \alpha(\sigma(t_1'), \sigma(t_2'))$ with the atoms:

$$\alpha_{s_0,s_1}(\sigma(t_1'), y), \alpha_{s_1,s_2}(y, y), \alpha_{s_{n-2},s_{n-1}}(y, y), \alpha_{s_{n-1},s_n}(y, \sigma(t_2')).$$

The final choices to be made occur in Step 5, where we must choose a concept $D \in \mathsf{BC}_{\mathcal{T}}$ and role $R \in \mathsf{N}_\mathsf{R}^\pm$ such that conditions (a), (b), and (c) are satisfied. We set $R = S$ (recall that $\pi(y) = eSC$). If $e \notin \mathsf{Ind}(\mathcal{A})$, then we let D be the unique concept such that $e = e'PD$. Note that if we are in DL-Lite$_\mathcal{R}$, then $D = \exists P^-$. It follows from the definition of canonical models that $\mathcal{T} \models D \sqsubseteq \exists S$ (if we are in DL-Lite$_\mathcal{R}$) or $\mathcal{T} \models D \sqsubseteq \exists S.C$ (for $\mathcal{ELH}$), so condition (a) holds. If instead we have $e \in \mathsf{Ind}$, then the definition of canonical models, together with our normal form for $\mathcal{ELH}$ TBoxes, guarantees that there is some $D \in \mathsf{BC}_{\mathcal{T}}$ such that $e \in D^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and $\mathcal{T} \models D \sqsubseteq \exists S$ (if we are in DL-Lite$_\mathcal{R}$) or $\mathcal{T} \models D \sqsubseteq \exists S.C$ (for $\mathcal{ELH}$). Note that in the case of DL-Lite$_\mathcal{R}$, $\mathcal{T}, \mathcal{A} \models \exists S(e)$ implies that one of the following holds: (i) there is a concept assertion $A(e) \in \mathcal{A}$ such that $\mathcal{T} \models A \sqsubseteq \exists S$, (ii) there is a role assertion $S'(e, e') \in \mathcal{A}$ such that $\mathcal{T} \models \exists S' \sqsubseteq \exists S$, or (iii) there is a role assertion $S'(e', e) \in \mathcal{A}$ such that $\mathcal{T} \models \exists(S')^- \sqsubseteq \exists S$. Thus, it is always possible to choose D such that $\mathcal{A} \models D(e)$, and we will assume in what follows that D has this property.

It remains to show that conditions (b) and (c) are verified when $R = S$. For (b), consider some binary atom $\beta(y, t)$ which belongs to the query at the start of Step 5. Then we know that there must exist an atom $\alpha(t_1', t_2') \in q$ with $\alpha = (S, \Sigma, \delta, s_0, F)$ such that $\beta(y, t)$ is equal to one of the following atoms which replaced $\alpha(\sigma(t_1'), \sigma(t_2'))$ during Step 3: $\alpha_{s_0,s_1}(\sigma(t_1'), y), \alpha_{s_1,s_2}(y, y), \ldots, \alpha_{s_{n-2},s_{n-1}}(y, y), \alpha_{s_{n-1},s_n}(y, \sigma(t_2'))$. Thus, we have an atom of the form $\alpha_{s_{i-1},s_i}(y, t)$. Using property $(\star)$ and considering the different possible values for t, we can infer that $(\pi(y), \pi(t)) \in L(\alpha_{s_{i-1},s_i})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, as witnessed by the path $p_i =$

$e_{j_{i-1}} u_{j_{i-1}+1} \dots u_{j_i} e_{j_i}$. We also know that $e_\ell \neq \pi(y)$ for every $j_{i-1} < \ell < j_i$. In particular, this means that either the path p_i is entirely contained in the subtree rooted at $\pi(y)$ or it never visits any element below $\pi(y)$. The former option cannot hold, since it would imply that $t = y$ and that $C \in \mathsf{Loop}_\alpha[s_i, s_{i+1}]$, so the atom would have been removed in Step 4. Thus, it must be the case that the first "step" in the path p_i goes from $\pi(y)$ to its parent e. It follows that $u_{j_{i-1}+1} = U^-$ for some $U \in \mathsf{N}_\mathsf{R}^\pm$ such that $(e, \pi(y)) \in U^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Since $\pi(y) = eSC$, we must have $\mathcal{T} \models S \sqsubseteq U$. Since $\lambda(p_i) \in L(\alpha_{s_{i-1},s_i})$, there must exist a state $s' \in S$ such that $(s_{i-1}, U^-, s') \in \delta$ and $u_{j_{i-1}+2} \dots u_{j_i} \in L(\alpha_{s',s_i})$. This shows that condition (b) is satisfied, and also that $(e, \pi(t)) \in L(\alpha_{s',s})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$

We now consider condition (c). Take some atom of the form $\beta(t, y)$ which appears in the query at the start of Step 5. Then we know from earlier that we can find some atom $\alpha(t'_1, t'_2) \in q$ (where $\alpha = (S, \Sigma, \delta, s_0, F)$) such that $\beta(t, y)$ is equal to one of the following atoms that replaced $\alpha(\sigma(t'_1), \sigma(t'_2))$ during Step 3: $\alpha_{s_0,s_1}(\sigma(t'_1), y)$, $\alpha_{s_1,s_2}(y, y)$, $\dots, \alpha_{s_{n-2},s_{n-1}}(y, y)$, $\alpha_{s_{n-1},s_n}(y, \sigma(t'_2))$. It follows that $\beta(t, y)$ has the form $\alpha_{s_i,s_{i+1}}(t, y)$. Using property ($\star$), and considering the two possible values for t, we can deduce that $(\pi(t), \pi(y)) \in L(\alpha_{s_i,s_{i+1}})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, as witnessed by the path $p_i = e_{j_{i-1}} u_{j_{i-1}+1} \dots u_{j_i} e_{j_i}$. Arguing as we did for (b), we can show that the path p_i is entirely contained in the subtree rooted at $\pi(y)$ or the path never visits any element below $\pi(y)$. The former option would imply that $t = y$ and that $C \in \mathsf{Loop}_\alpha[s_i, s_{i+1}]$, so the atom would have been removed in Step 5. Thus, it must be the case that the last "step" in the path is from e to $\pi(y)$, so $u_{j_i} = U$ for some $U \in \mathsf{N}_\mathsf{R}^\pm$ such that $(e, \pi(y)) \in U^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Since $\pi(y) = eSC$, we must have $\mathcal{T} \models S \sqsubseteq U$. Since $\lambda(p_i) \in L(\alpha_{s_i,s_{i+1}})$, we also know that there must exist a state s'' such that $(s'', U, s_{i+1}) \in \delta$ and $u_{j_{i-1}+1} \dots u_{j_i-1} \in L(\alpha_{s_i,s''})$. This shows that condition (c) is satisfied, and also that $(\pi(t), e) \in L(\alpha_{s_i,s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. It is also important to note that if $t = y$, then we can apply the arguments for conditions (b) and (c) together to show that $(e, e) \in L(\alpha_{s',s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ (with state s' as in (b), and s'' as required for (c)).

Now let q' be the query we obtain at the end of Step 7 when all non-deterministic choices are made in the manner that we have described. Consider the mapping π' defined as follows:

- $\pi'(t) = \pi(t)$ for every $t \in \mathsf{terms}(q)$ with $\pi(t) \neq \pi(y)$.
- $\pi'(t) = e$ for every $t \in \mathsf{terms}(q)$ with $\pi(t) = \pi(y)$.

Note that the mapping π' satisfies the properties (c1) and (c2). It remains to show that π' is a match for q'.

Consider first a concept atom $B(t) \in q'$. There are two possibilities. Either $B(t)$ appears in q and $t \notin \mathsf{Leaf}$, or $B(t)$ was introduced in Step 7. In the former case, we know that π satisfies $B(t)$, and since $\pi'(t) = \pi(t)$ (since $t \notin \mathsf{Leaf}$), the same is true of π'. In the latter case, we must have $t = y$ and either $B = D$ if $D \in \mathsf{N}_\mathsf{C}$ or $B = A_{P^-}$ if $D = \exists P^-$. If $B = D$, then we can use the fact that $\pi'(t) = e$ and D was chosen so that $e \in D^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ to infer that π' satisfies $B(t)$. If $B = A_{P^-}$, then we have that $e \in (\exists P^-)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Since $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is a model of $\mathcal{T}$ and $\exists P^- \sqsubseteq A_{P^-} \in \mathcal{T}$, we also have $e \in A_{P^-}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, which means that π' satisfies $B(t)$.

Now consider an atom $\gamma(t', t'') \in q'$. If $y \notin \{t', t''\}$, then $\gamma(t', t'') \in q$. As π is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, it must be the case that $(\pi(t'), \pi(t'')) \in L(\gamma)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Since $\pi'(t') = \pi(t')$ and $\pi'(t'') = \pi(t'')$, the same holds for π', and so the atom $\gamma(t', t'')$ is satisfied by π'. Next suppose that $y \in \{t', t''\}$. An examination of $\mathsf{Rewrite}$ shows that $\gamma(t', t'')$ must have replaced some atom during Step 6. We distinguish three cases:

- Case 1: $\gamma(t', t'')$ replaces $\alpha_{s_i, s_{i+1}}(y, t)$ with $t \neq y$. Then $\gamma(t', t'')$ must have the form $\alpha_{s', s_{i+1}}(y, t)$, where s' is the state that was chosen to ensure condition (b) in Step 5. We recall that s' is such that $(e, \pi(t)) \in L(\alpha_{s', s_{i+1}})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Since $t \neq y$, we know that $t \notin \mathsf{Leaf}$, and so $\pi(t) = \pi'(t)$. It follows that $(\pi'(y), \pi'(t)) \in L(\alpha_{s', s_{i+1}})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, so the atom $\gamma(t', t'')$ is satisfied by π'.

- Case 2: $\gamma(t', t'')$ replaces $\alpha_{s_i, s_{i+1}}(t, y)$ with $t \neq y$. Then $\gamma(t', t'')$ must have the form $\alpha_{s_i, s''}(t, y)$, where s'' is the state that was used in condition 5(c). We showed earlier when examining condition (c) that $(\pi(t), e) \in L(\alpha_{s_i, s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Using the fact that $\pi'(t) = \pi(t)$ and $\pi'(y) = e$, we can infer that $(\pi'(t), \pi'(y)) \in L(\alpha_{s_i, s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, hence π' satisfies the atom $\gamma(t', t'')$.

- Case 3: $\gamma(t', t'')$ replaces $\alpha_{s_i, s_{i+1}}(y, y)$. Then $\gamma(t', t'')$ must have the form $\alpha_{s', s''}(y, y)$, where s' is the state from 5(b) and s'' is the state from 5(c). We have $(\pi'(y), \pi'(y)) = (e, e) \in L(\alpha_{s', s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, which means that π' satisfies the atom $\gamma(t', t'')$.

As we have shown that every atom in q' is satisfied by the mapping π', it follows that π' is a match for q' in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, which completes the proof. ☐

Lemma 6.5. *If $\vec{a} \in \mathsf{cert}(q', (\mathcal{T}, \mathcal{A}))$ for some $q' \in \mathsf{Rewrite}(q, \mathcal{T})$, then $\vec{a} \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$.*

Proof. It is sufficient to show that if $q' \in \mathsf{OneStep}(q, \mathcal{T})$ and $\vec{a} \in \mathsf{cert}(q', (\mathcal{T}, \mathcal{A}))$, then $\vec{a} \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$. Fix a C2RPQ q and a DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ TBox $\mathcal{T}$, and let $q' \in \mathsf{OneStep}(q, \mathcal{T})$ be such that $\vec{a} \in \mathsf{cert}(q', (\mathcal{T}, \mathcal{A}))$. By Lemma 3.2, there exists a match π' for q' in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi'(\vec{x}) = \vec{a}$, with $\vec{x}$ the answer variables of q'.

Consider the execution of $\mathsf{OneStep}(q, \mathcal{T})$ that leads to the query q' being output. Let Leaf be the non-empty subset of $\mathsf{qvars}(q)$ that was selected in Step 1, let y be the variable from Leaf chosen in Step 1, let $C \in \mathsf{TC}_\mathcal{T}$ be the concept selected in Step 2, and let $D \in \mathsf{BC}_\mathcal{T}$ and R be the concept and role selected in Step 5. In Step 7, if $D \in \mathsf{N_C}$, then $D(y)$ was added, and if $D = \exists P^-$, then $A_{P^-}(y)$ was added. In the former case, we know that $\pi'(y) \in D^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and that either (i) $\mathcal{T} \models D \sqsubseteq \exists R$ and $C = \exists R^-$, or (ii) $\mathcal{T} \models D \sqsubseteq \exists R.C$, hence there must exist an R-child e of $\pi'(y)$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ with $\mathsf{Tail}(e) = C$. In the latter case, we have $\pi'(y) \in A_{P^-}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. By our assumption on DL-Lite$_\mathcal{R}$ TBoxes, $\mathcal{T}$ must contain the inclusion $A_{P^-} \sqsubseteq \exists P^-$. Since $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is a model of $\mathcal{T}$, this yields $\pi'(y) \in (\exists P^-)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, and hence $\pi'(y) \in D^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. We then use the fact that $\mathcal{T} \models D \sqsubseteq \exists R$ where $C = \exists R^-$ to find an R-child e of $\pi'(y)$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ with $\mathsf{Tail}(e) = C$.

We define a mapping $\pi : \mathsf{terms}(q) \to \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ by setting $\pi(t) = e$ for every $t \in \mathsf{Leaf}$ and setting $\pi(t) = \pi'(t)$ for every $t \in \mathsf{terms}(q') \setminus \{y\}$. This mapping is well-defined since every term in q either belongs to Leaf or appears in q'. Observe that $\pi(\vec{x}) = \vec{a}$ since $\pi'(\vec{x}) = \vec{a}$ and Leaf does not contain answer variables. We aim to show that π is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. To this end, consider some concept atom $B(t) \in q$. First suppose that $t \in \mathsf{Leaf}$. Then we know that the concept C selected in Step 2 is such that $\mathcal{T} \models C \sqsubseteq B$. We then use the fact that since $t \in \mathsf{Leaf}$, we have $\pi(t) = e \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. If $t \notin \mathsf{Leaf}$, then $B(t) \in q'$. As π' is a match for q', we have $\pi'(t) \in B^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Since $\pi'(t) = \pi(t)$, we get $\pi(t) \in B^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

Now consider some atom of the form $\alpha(t, t') \in q$, where $\alpha = (S, \Sigma, \delta, s_0, F)$. If both $t \notin \mathsf{Leaf}$ and $t' \notin \mathsf{Leaf}$, then it can be verified that $\alpha(t, t') \in q'$. As π' is a match for q' in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, it must be the case that $(\pi'(t), \pi'(t')) \in L(\alpha)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Since $\pi'(t) = \pi(t)$ and $\pi'(t) = \pi(t)$, the same holds for π. Let us next consider the more interesting case in which $\{t, t'\} \cap \mathsf{Leaf} \neq \emptyset$. In Step 3, we have a query containing $\alpha(\sigma(t), \sigma(t'))$, where the mapping σ is defined as follows: $\sigma(t'') = t''$ for $t'' \notin \mathsf{Leaf}$ and $\sigma(t'') = y$ for $t'' \in \mathsf{Leaf}$. Note that since $\{t, t'\} \cap \mathsf{Leaf} \neq \emptyset$, at least one of $\sigma(t)$ and $\sigma(t')$ must be y. It follows that in Step 3, we will guess a sequence $s_1, \ldots, s_{n-1}$ of distinct states from S and a state $s_n \in F$, and we will replace $\alpha(\sigma(t), \sigma(t'))$ by the atoms: $\alpha_{s_0,s_1}(\sigma(t), y)$, $\alpha_{s_1,s_2}(y, y)$, $\ldots$, $\alpha_{s_{n-2},s_{n-1}}(y, y)$, $\alpha_{s_{n-1},s_n}(y, \sigma(t'))$. Let us denote this set of atoms by Q_α. Slightly abusing terminology, we use the phrase *match for* Q_α to refer to a match for the Boolean query given by the conjunctions of all atoms in Q_α. We now establish the following claim:

Claim 1. If π is a match for Q_α in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, then π is a match for $\alpha(t, t')$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$.

Proof of claim. Suppose that π is a match for the atoms in Q_α in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. This means that

- $(\pi(\sigma(t)), \pi(y)) \in L(\alpha_{s_0,s_1})\mathcal{I}_{\mathcal{T},\mathcal{A}}$,
- $(\pi(y), \pi(y)) \in L(\alpha_{s_i,s_{i+1}})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ for every $1 \leq i < n-1$, and
- $(\pi(y), \pi(\sigma(t'))) \in L(\alpha_{s_{n-1},s_n})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

We then remark that the language consisting of all words $w_1 \ldots w_n$ such that $w_1 \in L(\alpha_{s_0,s_1})$, $w_i \in L(\alpha_{s_i,s_{i+1}})$ for every $1 \leq i < n-1$, and $w_n \in L(\alpha_{s_{n-1},s_n})$ is a subset of the language $L(\alpha_{s_0,s_n})$, and hence of $L(\alpha)$. Thus, by composing the paths witnessing each of the statements in the preceding list, we can show that $(\pi(\sigma(t), \pi(\sigma(t')) \in L(\alpha)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Then to complete the proof, we simply note that $\pi(\sigma(t)) = \pi(t)$ and $\pi(\sigma(t')) = \pi(t)$, because of the way we defined π and σ. (*end proof of claim*)

Because of Claim 1, to complete the proof that π is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, it is sufficient to show that π is a match for Q_α, which is established by the following claim:

Claim 2. For every $\alpha_{s,s'}(u, u') \in Q_\alpha$: $(\pi(u), \pi(u')) \in L(\alpha_{s,s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

Proof of claim. Take some $\alpha_{s,s'}(u, u') \in Q_\alpha$. We start with the case where $u = u' = y$ and $C \in \mathsf{Loop}_\alpha[s, s']$. As $\pi(y) = e$ and $\mathsf{Tail}(e) = C$, we have $\mathsf{Tail}(\pi(y)) = C$. We can thus apply Proposition 5.5 to infer that $(\pi(y), \pi(y)) \in L(\alpha_{s,s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, which yields the desired result given that $u = u' = y$. Next suppose that either $u \neq y$, $u' \neq y$, or $C \notin \mathsf{Loop}_\alpha[s, s']$. Then we will not remove $\alpha_{s,s'}(u, u')$ in Step 4, so it will still be present in Step 5. There are three cases depending on which of u and u' equals y. We treat each case separately:

- Case 1: $u = y$ and $u' \neq y$. It follows that $\pi(u) = e$ and $\pi(u') = \pi'(u')$. In Step 6, we will replace $\alpha_{s,s'}(u, u')$ with $\alpha_{s'',s'}(u, u')$ where $s'' \in S$ is such that $(s, U^-, s'') \in \delta$ for some $U \in \mathsf{N}_\mathsf{R}^\pm$ with $\mathcal{T} \models R \sqsubseteq U$. Using the facts that $(\pi'(y), e) \in R^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, $\pi(u) = e$, $\mathcal{T} \models R \sqsubseteq U$, and $(s, U^-, s'') \in \delta$, we can infer that $(\pi(u), \pi'(y)) \in L(\alpha_{s,s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. We also know that the atom $\alpha_{s'',s'}(u, u')$ belongs to q', and so it must be satisfied by π', which yields$(\pi'(u), \pi'(u')) \in L(\alpha_{s'',s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. By combining $(\pi(u), \pi'(y)) \in L(\alpha_{s,s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and $(\pi'(u), \pi'(u')) \in L(\alpha_{s'',s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, and using the fact that $(\pi'(u), \pi'(u')) = (\pi'(y), \pi(u'))$, we can infer that $(\pi(u), \pi(u')) \in L(\alpha_{s,s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

- Case 2: $u \neq y$ and $u' = y$. It follows that $\pi(u) = \pi'(u)$ and $\pi(u') = e$. In Step 6, we will replace $\alpha_{s,s'}(u, u')$ with an atom $\alpha_{s,s''}(u, u')$ where $s'' \in S$ is such that $(s'', U, s') \in \delta$ for some $U \in \mathsf{N}_\mathsf{R}^\pm$ with $\mathcal{T} \models R \sqsubseteq U$. Using similar arguments to Case 1, we can show that $(\pi(u), \pi'(y)) \in L(\alpha_{s,s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and $(\pi'(y), \pi(u')) \in L(\alpha_{s'',s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, from which we can deduce that $(\pi(u), \pi(u')) \in L(\alpha_{s,s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

- Case 3: $u = u' = y$. It follows that $\pi(u) = \pi(u') = e$. In Step 6, we will replace $\alpha_{s,s'}(u, u')$ with an atom $\alpha_{s'',s'''}(u, u')$ where s'', s''' are such that $(s, U^-, s'') \in \delta$ and $(s'', U', s') \in \delta$ for some roles $U, U' \in \mathsf{N}_\mathsf{R}^\pm$ with $\mathcal{T} \models R \sqsubseteq U$ and $\mathcal{T} \models R \sqsubseteq U'$. By applying similar reasoning to that used in Cases 1 and 2, we can show that $(\pi(u), \pi'(y)) \in L(\alpha_{s,s''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, $(\pi'(y), \pi'(y)) \in L(\alpha_{s'',s'''})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, and $(\pi'(y), \pi(u')) \in L(\alpha_{s''',s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. From this, we can infer that $(\pi(u), \pi(u')) \in L(\alpha_{s,s'})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. □

Together, Lemmas 6.4 and 6.5 establish Proposition 6.3. We remark that the number of possible atoms appearing in the queries in $\mathsf{Rewrite}(q, \mathcal{T})$ is polynomially bounded by $|q| + |\mathcal{T}|$. This is the key property used to show the following:

Proposition 6.6. *There are only exponentially many queries in* $\mathsf{Rewrite}(q, \mathcal{T})$*, each having size polynomial in* $|q| + |\mathcal{T}|$*.*

Proof. Consider a DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ TBox $\mathcal{T}$, a C2RPQ q, and $q' \in \mathsf{Rewrite}(q, \mathcal{T})$. We first note that $\mathsf{OneStep}$ never introduces any fresh variables, so $\mathsf{vars}(q') \subseteq \mathsf{vars}(q)$. We next note that $\mathsf{OneStep}$ does not introduce any fresh concept names, and it only introduces binary atoms whose NFAs are obtained from one of the original NFAs by changing the initial and finite states. Thus, every atom in q' takes one of the following forms:

- $A(v)$, where $A \in \mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})$ and $v \in \mathsf{vars}(q)$
- $\alpha_{s,s'}(v, v')$, where α appears in q, s, s' are states of α, and $v, v' \in \mathsf{vars}(q)$

It is easy to see that the number of such atoms is bounded polynomially in $|\mathcal{T}| + |q|$, and thus, there are at most single-exponentially many distinct queries in $\mathsf{Rewrite}(q, \mathcal{T})$. □

6.2 Query Evaluation

In Figure 16, we present a non-deterministic algorithm $\mathsf{EvalQuery}$ for C2RPQ answering in DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$. The algorithm starts by checking whether the input KB is satisfiable. If the check succeeds, then the algorithm guesses a query from $\mathsf{Rewrite}(q, \mathcal{T})$ and a variable assignment and calls the evaluation algorithm[3] $\mathsf{EvalAtom}$ from Section 5 to check whether this assignment yields a match for this query in $\mathcal{I}_\mathcal{K}$. The following proposition establishes the correctness of $\mathsf{EvalQuery}$.

Proposition 6.7. *For every C2RPQ q, DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and tuple of individuals $\vec{a}$ from* $\mathsf{Ind}(\mathcal{A})$ *of same arity as q: $\vec{a} \in$* $\mathsf{cert}(q, \mathcal{K})$ *if and only if there is some execution of* $\mathsf{EvalQuery}(q, \mathcal{K}, \vec{a})$ *that returns* yes*.*

3. As we check KB satisfiability in the first step of $\mathsf{EvalQuery}$, we may skip the satisfiability checks in the calls to $\mathsf{EvalAtom}$.

```
ALGORITHM EvalQuery
Input: C2RPQ $q(x_1, \ldots, x_k)$, DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, tuple $\vec{a} \in \mathsf{Ind}(\mathcal{A})^k$

1. Test whether $\mathcal{K}$ is satisfiable, output yes if not.

2. Guess some $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ and an assignment $\vec{b}$ of individuals to $\mathsf{qvars}(q')$.

    (a) Let $q''$ be the query obtained by substituting $\vec{a}$ for $(x_1, \ldots, x_k)$ and $\vec{b}$ for $\mathsf{qvars}(q')$,
        then replacing each atom of the form $B(a)$ by the atom $\alpha_{B?}(a)$ where $\alpha_{B?}$ is an
        NFA with $L(\alpha_{B?}) = \{B?\}$ consisting on an initial state $s_0^B$, a single final state
        $s_f^B$, and a single transition $(s_0^B, B?, s_f^B)$.
    (b) For every atom $\alpha(a, b)$ in $q''$
            check if $\mathsf{EvalAtom}(\alpha, \mathcal{K}, (a, b)) =$ yes
    (c) If all checks succeed, return yes.

3. Return no.
```

Figure 16: Non-deterministic C2RPQ answering algorithm EvalQuery.

Proof. Let $\vec{x}$ be the set of answer variables of q.

To show the first direction, consider an execution of EvalQuery on input $(q, \mathcal{K}, \vec{a})$ that returns yes. If the algorithm returns yes in Step 1, then $\mathcal{K}$ is unsatisfiable, so we trivially have $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$. Otherwise, in Step 2, the algorithm will guess a query $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ and assignment $\vec{b}$ for the quantified variables $\vec{y}$ of q'. Let q'' be the query obtained by substituting $\vec{a}$ for $\vec{x}$ and $\vec{b}$ for $\vec{y}$, and writing all atoms in the form $\alpha(a, b)$ with α an NFA. In Step 2(c), for every atom $\alpha(a, b)$ in q'', we call EvalAtom on input $(\alpha, \mathcal{K}, (a, b))$. Since the algorithm returns yes in Step 3, it must be the case that all of these calls return yes, and so by Proposition 5.8, $(a, b) \in \mathsf{cert}(q, \mathcal{K})$ for every atom $\alpha(a, b)$ in q''. It follows that the mapping π sending $\vec{x}$ to $\vec{a}$ and $\vec{y}$ to $\vec{b}$ defines a match for q' in $\mathcal{I}_\mathcal{K}$, so $\vec{a} \in \mathsf{cert}(q', \mathcal{K})$. Applying Proposition 6.3, we obtain $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$.

Next suppose that $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$. If $\mathcal{K}$ is unsatisfiable, then the algorithm will return yes in Step 1. Otherwise, we know from Proposition 6.3 that there exists a query $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ and a match π for q' in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ that maps all variables to $\mathsf{Ind}(\mathcal{A})$ and is such that $\pi(\vec{x}) = \vec{a}$. In Step 2, we choose the query q' and tuple $\pi(\vec{y})$, where $\vec{y}$ is the set of quantified variables in q'. Let q'' be the query obtained by substituting $\vec{a}$ for $\vec{x}$ and $\vec{b}$ for $\vec{y}$, and writing all atoms in the form $\alpha(a, b)$ with α an NFA. Because of the match π we know that $(a, b) \in \mathsf{cert}(q, \mathcal{K})$ for every atom $\alpha(a, b)$ in q''. It follows then by Proposition 5.8 that all of the calls to EvalAtom will return yes, and EvalQuery will return yes in Step 2(c). □

By analyzing the complexity of the algorithm EvalQuery, we obtain the following upper bounds for C2RPQ answering, which match the lower bounds given in Section 4.

Theorem 6.8. *C2RPQ answering is in*

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. For Statement (1), we consider the resources required to run $\mathsf{EvalQuery}$ on input $(\mathcal{T}, \mathcal{A}, q)$, where $\mathcal{T}$ is formulated in DL-Lite$_{\mathcal{R}}$. The consistency check in Step 1 can be performed in non-deterministic logarithmic space in $|\mathcal{A}|$ (Calvanese et al., 2007). If $\mathcal{T}$ and q are treated as fixed, then computing $\mathsf{Rewrite}(q, \mathcal{T})$ requires only constant time (and space) in $|\mathcal{A}|$. It follows that the guessed query q' and tuple $\vec{b}$ and the set of atoms at the end of Step 2(a) can be stored using logarithmic space in $|\mathcal{A}|$. In Step 2(b), we call $\mathsf{EvalAtom}$ on each of the stored atoms. By Theorem 5.9, $\mathsf{EvalAtom}$ runs in non-deterministic logarithmic space in $|\mathcal{A}|$. Since $\mathrm{NL}^{\mathrm{NL}} = \mathrm{NL}$, we obtain the desired NL upper bound in data complexity.

To show Statement 2, consider what happens if the input TBox $\mathcal{T}$ is formulated in $\mathcal{ELH}$. In this case, the consistency check in Step 1 takes constant time (since every $\mathcal{ELH}$ KB is satisfiable), and by Theorem 5.9, $\mathsf{EvalAtom}$ runs in polynomial time in $|\mathcal{A}|$. We thus have a decision procedure for C2RPQ answering in $\mathcal{ELH}$ that runs in non-deterministic logarithmic space in $|\mathcal{A}|$ with access to a P oracle. Since $\mathrm{NL}^{\mathrm{P}} = \mathrm{P}$, this yields membership in P for data complexity.

To establish Statement (3), we first note that if $\mathcal{T}$ is a DL-Lite$_{\mathsf{RDFS}}$ TBox, then the query cannot be rewritten, i.e., $\mathsf{Rewrite}(q, \mathcal{T}) = \{q\}$ (we cannot choose a D as required in Step 5 of Algorithm $\mathsf{OneStep}$ because $\mathcal{T}$ does not entail any inclusions of the form $D \sqsubseteq \exists R$). Thus, in Step 2 of $\mathsf{EvalQuery}$, we only need to guess a tuple $\vec{b}$ whose size is polynomial in $|\mathcal{A}|$ and $|q|$. The calls to $\mathsf{EvalAtom}$ in Step 2(b) run in polynomial time in the input (Theorem 5.2), so the overall procedure runs in NP.

For Statement (4), instead of computing the whole set $\mathsf{Rewrite}(q, \mathcal{T})$, which can contain exponentially many queries, we generate a single $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ non-deterministically. By Proposition 6.6, 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 (Proposition 6.6), 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 proceed as for statement 3, guessing a (polynomial-size) tuple $\vec{b}$ and performing a polynomial number of polynomial-time evaluation checks (Theorem 5.9). This yields a non-deterministic polynomial space procedure for deciding $\vec{a} \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$. Using the well-known fact that NPSpace = PSpace, we obtain the desired PSpace upper bound. □

6.3 Cases with Lower Complexity

Given the substantial jump in combined complexity – from NP to PSpace – when moving from CQs to C(2)RPQs, it is natural to look for interesting subcases that offer lower complexity. We pinpoint two such subcases, the first obtained by restricting the query language, and the second obtained by restricting the class of KBs.

Let us recall that 2RPQs are single-atom C2RPQs that do not contain quantified variables. The following theorem shows that this restriction is inessential, as our complexity results for 2RPQs hold also for single-atom queries with quantified variables.[4]

Theorem 6.9. *The results in Theorem 5.9 hold also for single-atom C2RPQs.*

Proof. Fix a DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. There are six types of single-atom queries to consider: $\exists y.\, \alpha(a, y)$ (with a an individual), $\exists y.\, \alpha(x, y)$ (with x an answer variable), $\exists y.\, \alpha(y, a)$ (with a an individual), $\exists y.\, \alpha(y, x)$ (with x an answer variable), $\exists x, y.\, \alpha(x, y)$ (with $x \neq y$), and $\exists y.\, \alpha(y, y)$. For the first five types of queries, there are simple reductions to 2RPQs. When $q = \exists x, y.\, \alpha(x, y)$ (with $x \neq y$), we can simply replace q by the 2RPQ $q'(x, y) = \alpha'(x, y)$, where

$$L(\alpha') = (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^* \cdot L(\alpha) \cdot (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^*.$$

The following claim establishes the correctness of this reduction.

Claim. $\mathcal{K} \models q$ if and only if $\mathsf{cert}(q', \mathcal{K}) \neq \emptyset$.

Proof of claim. First suppose that $\mathcal{K} \models q$. If $\mathcal{K}$ is unsatisfiable, then we trivially have $\mathsf{cert}(q', \mathcal{K}) \neq \emptyset$. Otherwise, there is a match π for q in $\mathcal{I}_\mathcal{K}$. This means that $(\pi(x), \pi(y)) \in L(\alpha)^{\mathcal{I}_\mathcal{K}}$, and so there must exist some path p_0 from $\pi(x)$ to $\pi(y)$ with $\lambda(p_0) \in L(\alpha)$. Let a be such that $\pi(x)$ is either equal to a or begins by a, and let p_1 be any path from a to $\pi(x)$. Since $\pi(y)$ is reachable from $\pi(x)$, it must also be reachable from a, and so we can find a path p_2 from $\pi(y)$ to a. We may choose p_1 and p_2 so that $\lambda(p_1)$ and $\lambda(p_2)$ belong to $(\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^*$. By combining the paths p_1, p_0 and p_2 (in that order), we obtain a path from a to a whose label belongs to $(\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^* \cdot L(\alpha) \cdot (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^*$. It follows that $(a, a) \in \mathsf{cert}(q', \mathcal{K})$.

Suppose next that $(a, b) \in \mathsf{cert}(q', \mathcal{K})$ for two (not necessarily distinct) individuals a, b. If $\mathcal{K}$ is unsatisfiable or $\varepsilon \in L(\alpha)$, then we trivially have $\mathcal{K} \models q$. Otherwise, there is a path $p = e_0 u_1 e_1 u_2 \ldots u_n e_n$ in $\mathcal{I}_\mathcal{K}$ with $e_0 = a$, $e_n = b$, and $\lambda(p) \in (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^* \cdot L(\alpha) \cdot (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^*$. Since $\varepsilon \notin L(\alpha)$, there exists $0 < i < j \leq n$ such that $u_i \ldots u_{j-1} \in L(\alpha)$. By setting $\pi(x) = e_{i-1}$ and $\pi(y) \in e_j$, we obtain a match for q in $\mathcal{I}_\mathcal{K}$. (*end proof of claim*)

The other four types of queries containing distinct terms can be handled similarly:

- For $\exists y.\, \alpha(x, y)$, we use the 2RPQ $q''(x, y) = \alpha''(x, y)$, where $L(\alpha'') = L(\alpha) \cdot (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^*$. Arguing as in the claim, we can show that $b \in \mathsf{cert}(\exists y.\, \alpha(x, y), \mathcal{K})$ iff $(b, c) \in \mathsf{cert}(q''(x, y), \mathcal{K})$ for some $c \in \mathsf{Ind}(\mathcal{A})$.

- For $\exists y.\, \alpha(a, y)$, we have $\mathcal{K} \models \exists y.\, \alpha(a, y)$ iff $a \in \mathsf{cert}(\exists y.\, \alpha(x, y), \mathcal{K})$, so we can reuse the 2RPQ from the preceding point.

- For $\exists y.\, \alpha(y, x)$, we use the 2RPQ $q'''(x, y) = \alpha'''(y, x)$ with $L(\alpha''') = (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^* \cdot L(\alpha)$. Using a similar argument to that used in the preceding claim, we can show that $b \in \mathsf{cert}(\exists y.\, \alpha(y, x), \mathcal{K})$ iff $(b, c) \in \mathsf{cert}(q'''(x, y), \mathcal{K})$ for some $c \in \mathsf{Ind}(\mathcal{A})$.

4. In the preliminary version of this paper, we in fact used the more general notion of single-atom queries (possibly with quantified variables) as the definition of 2RPQs (Bienvenu et al., 2013).

- For $\exists y.\, \alpha(y,a)$, we have $\mathcal{K} \models \exists y.\, \alpha(y,a)$ iff $a \in \mathsf{cert}(\exists y.\, \alpha(y,x), \mathcal{K})$, so we can reuse the 2RPQ from the preceding point.

For queries of the form $\exists x.\, \alpha(x,x)$, the proof is more involved and passes by the definition of an alternative rewriting procedure for 2RPQs, which is similar in spirit to the algorithm Rewrite but guaranteed to run in polynomial time. Details are given in the appendix. $\square$

More interestingly, we can adapt the techniques from the preceding proof in order to provide an NP upper bound for the class of C2RPQ s that do not contain any *existential-join variables*, i.e., existentially quantified variables that occur more than once in a query.

Theorem 6.10. *C2RPQ answering is in* NP *in combined complexity for DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ knowledge bases when restricted to C2RPQs without existential-join variables.*

Proof. Consider a DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a C2RPQ q with answer variables $\vec{x}$ and existential variables $\vec{y}$ such that every variable in $\vec{y}$ occurs exactly once in q. We let q' be the C2RPQ obtained from q as follows:

- Replace every atom $\alpha(y_1, y_2)$ such that y_1, y_2 are both existential variables by the atom $\alpha'(y_1, y_2)$ where $L(\alpha') = (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^* \cdot L(\alpha) \cdot (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^*$.
- Replace every atom $\alpha(t,y)$ such that y is an existential variable and t is an individual or answer variable by the atom $\alpha''(t,y)$ where $L(\alpha'') = L(\alpha) \cdot (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^*$.
- Replace every atom $\alpha(y,t)$ such that y is an existential variable and t is an individual or answer variable by the atom $\alpha'''(y,t)$ where $L(\alpha''') = (\mathsf{N}_\mathsf{R}^\pm \cap \mathsf{sig}(\mathcal{T}))^* \cdot L(\alpha)$.
- Add $\vec{y}$ to the set of answer variables.

Clearly, it takes only polynomial time to construct the C2RPQ q'. By exploiting the fact that every existential variable in q occurs at most once, and applying similar reasoning to that used in the proof of Theorem 6.9, we can show that $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$ iff $(\vec{a}, \vec{b}) \in \mathsf{cert}(q', \mathcal{K})$ for some tuple of individuals $\vec{b}$ of the same arity as $\vec{y}$. To decide whether the latter holds, we non-deterministically guess a tuple $\vec{b}$ and let π be the variable assignment that maps $\vec{x}$ to $\vec{a}$ and $\vec{y}$ to $\vec{b}$. We then use EvalAtom to verify that $(\pi(t_1), \pi(_2)) \in \mathsf{cert}(\alpha(t_1,t_2), \mathcal{K})$ for every atom $\alpha(t_1,t_2) \in q'$, and we return `yes` if this is the case. Correctness of the described procedure follows from the correctness of EvalAtom (Proposition 5.8). Since EvalAtom can be implemented so as to run in polynomial time for both DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ (Theorem 5.9), we obtain an NP procedure for answering C2RPQs without existential-join variables. $\square$

The preceding result can be extended a bit further to allow for simple *chains* of existential variables. Indeed, we remark that if a C2RPQ q contains atoms $\alpha_1(x,y)$ and $\alpha_2(y,z)$ with z an existential-join variable and y an existential variable appearing only in these two atoms, then we can replace $\{\alpha_1(x,y), \alpha_2(y,z)\}$ by $\alpha_3(x,z)$ where $L(\alpha_3) = \{w_1 w_2 \mid w_1 \in L(\alpha_1), w_2 \in L(\alpha_2)\}$ (such an NFA α_3 is easily constructed from α_1, α_2 in polynomial time). By performing this and similar polynomial-time equivalence-preserving transformations, we can eliminate some existential-join variables and thereby enlarge the class of C2RPQs that can be handled using an NP procedure.

Finding further interesting classes of computationally well-behaved queries will likely prove difficult, given that the PSpace lower bound in Section 4 was shown to hold even under strong structural restrictions on C2RPQs. This suggests that it may be more fruitful to consider restrictions on knowledge bases. The next proposition identifies a natural restriction on knowledge bases that leads to an improved NP upper bound.

Theorem 6.11. *C2RPQ answering is in* NP *in combined complexity for DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ knowledge bases whose canonical models have finite domains.*

Proof. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a satisfiable DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ knowledge base whose canonical model $\mathcal{I}_\mathcal{K}$ contains only finitely many elements (note that if $\mathcal{K}$ is unsatisfiable, then it is trivial to perform query answering). It follows the construction of $\mathcal{I}_\mathcal{K}$ and the fact that $\Delta^{\mathcal{I}_\mathcal{K}}$ is finite that every element $aR_1C_1 \dots R_nC_n \in \Delta^{\mathcal{I}_\mathcal{K}}$ is such that $C_i \neq C_j$ for $i \neq j$, which implies in particular that $n \leq |\mathsf{TC}_\mathcal{T}|$ (indeed, if $C_i = C_j$ for $i < j$, then the domain of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ would contain the element $aR_1C_1 \dots R_jC_j(R_{i+1}C_{i+1} \dots R_jC_j)^m$ for every $m \geq 0$). We rely on this property to devise a non-deterministic algorithm for deciding $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$ that runs in polynomial time in the combined size of the inputs.

Without loss of generality, we may suppose that the input query q contains only binary atoms of the form $\alpha(t, t')$ with α an NFA. In a first step, we guess a mapping π from the terms in q to sequences of the form $aR_1C_1 \dots R_nC_n$ where each R_i is a role, each C_i is a concept from $\mathsf{TC}_\mathcal{T}$, and $0 \leq n \leq |\mathsf{TC}_\mathcal{T}|$. In a second step, we verify that π is indeed a match for q. First, we check that $\pi(b) = b$ for every individual b in q, and that $\pi(\vec{x}) = \vec{a}$ where $\vec{x}$ is the tuple of answer variables of q. Next, we check whether $\pi(z) \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ for every $z \in \mathsf{qvars}(q)$. This can be done with a polynomial number of (polynomial-time) entailment checks that verify conditions (A) and (B) of the definition of canonical models in Section 2.1.4. It only remains to check that all of the query atoms are satisfied under the mapping π. To this end, we construct a new ABox $\mathcal{A}_\pi$ as follows. Let E_π be the set containing each $aR_1C_1 \dots R_jC_j$ with $1 \leq j \leq n$ such that there is some $aR_1C_1 \dots R_jC_j \dots R_nC_n$ in the range of π. We introduce a fresh individual name b_e for each $e \in E_\pi$, and let $\mathcal{A}_\pi$ be the ABox obtained by adding to $\mathcal{A}$ the following assertions:

- $R(a, b_e)$ if $R \in \mathsf{N_R}$ and $R^-(b_e, a)$ if $R^- \in \mathsf{N_R}$ for each $e = aRC \in E_\pi$ where $a \in \mathsf{N_I}$
- $R(b_e, b_{e'})$ if $R \in \mathsf{N_R}$ and $R^-(b_{e'}, b_e)$ if $R^- \in \mathsf{N_R}$, for each $e' = eRC \in E_\pi$ where $e \in E_\pi$
- $C(b_{e'})$ for each $e' = eRC \in E_\pi$ where $e \in E_\pi \cup \mathsf{N_I}$ and $C \in \mathsf{N_C}$

Define the mapping π' from terms in q to individuals in $\mathcal{A}_\pi$ as follows: $\pi'(t) = \pi(t)$ if $\pi(t) \in \mathsf{Ind}(\mathcal{A})$ and $\pi'(t) = b_{\pi(t)}$ otherwise. For every atom $\alpha(t, t')$ in q, we call $\mathsf{EvalAtom}$ on input $(\alpha, (\mathcal{T}, \mathcal{A}_\pi), (\pi'(t), \pi'(t')))$; by Theorem 5.9, each of these calls needs only polynomial time. We output yes just in the case that every call to $\mathsf{EvalAtom}$ returns yes.

It should be clear that the algorithm just described runs in non-deterministic polynomial time. To show that the algorithm is sound, consider an execution of the algorithm that returns yes, and let π be the mapping that was guessed. Since the algorithm returned yes, we know that for every atom $\alpha(t, t')$ in q, there is a successful execution of $\mathsf{EvalAtom}$ on input $(\alpha, (\mathcal{T}, \mathcal{A}_\pi), (\pi'(t), \pi'(t')))$. Since $\mathsf{EvalAtom}$ is known to be correct (Theorem 6.7), this shows π' is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}_\pi}$. It follows from the way we defined $\mathcal{A}_\pi$ and the construction of canonical models that $\mathcal{I}_{\mathcal{T},\mathcal{A}_\pi}$ can be homomorphically embedded into $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. Moreover, we can choose the homomorphism h such that $h(b_e) = e$ for each of the new individuals b_e in $\mathcal{A}_\pi$. Since matches of C2RPQs are preserved under homomorphisms, it follows that π is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ with $\pi(\vec{x}) = \vec{a}$, so $\vec{a} \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$.

To show completeness, suppose that π is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. By construction, there is a homomorphism h from $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ to $\mathcal{I}_{\mathcal{T},\mathcal{A}_\pi}$ that maps every individual $a \in \mathsf{Ind}(\mathcal{A})$ to itself and every $e \in E_\pi$ to the individual b_e. Using the fact that query matches are preserved under homomorphisms, we have that π' is a match for q in $\mathcal{I}_{\mathcal{T},\mathcal{A}_\pi}$. It follows that for every atom $\alpha(t, t')$ in q, there is an execution of $\mathsf{EvalAtom}$ on $(\alpha, (\mathcal{T}, \mathcal{A}_\pi), (\pi'(t), \pi'(t')))$ that returns `yes`, and so the algorithm returns `yes` after guessing the mapping π. $\square$

We point out that the class of knowledge bases considered in Theorem 6.11 is of practical relevance. Indeed, several important large-scale $\mathcal{ELH}$ terminologies, like the medical ontology SNOMED[5], are *acyclic terminologies* (see, e.g., (Haase & Lutz, 2008)), which are guaranteed to have finite canonical models. Moreover, it has been recently argued that real-world DL-Lite$_\mathcal{R}$ ontologies often yield canonical models of shallow depth (Kikot, Kontchakov, Podolskii, & Zakharyaschev, 2013).

7. Beyond C2RPQs and Lightweight DLs

In this section, we discuss some of the implications of our results and give a brief overview of related results for similar settings.

7.1 Extensions of C2RPQs

We have argued in this paper that C2RPQs provide significantly more expressiveness than plain CQs as languages for querying ontologies, at a moderate computational cost. However, C2RPQs also have many limitations, and several application domains seem to call for even more expressive query languages. We now discuss some extensions to C2RPQs.

7.1.1 Complex Labels

In a preliminary version of the present work, we added to C2RPQs the ability to talk about combinations of concepts and roles that appear along a path (Bienvenu, Ortiz, & Šimkus, 2012). In that language, which we called *C2RPQs with complex labels*, one can use, for example, the expression $(\mathsf{sbLFT} \wedge \mathsf{sbLFSub})^*$ to find paths between stations that are served by both low-floor tramway and subway. Complex labels provide a more concise and flexible syntax for many queries, and in fact they increase the expressiveness of C2RPQs. In particular in DLs that do not support role conjunction, like the ones considered here, with standard C2RPQs we can query for pairs of stations that are connected by two different means of transport, but we cannot require the existence of one route between them that is fully served by both. All the algorithms described in this paper can be extended straightforwardly to C2RPQs with complex labels, and the given complexity results apply also to that more expressive query language. However, the extension to complex labels causes a significant overhead in the notation and the technicalities of the algorithms. Hence, for the sake of readability, we decided not to include such an extension in this paper.

5. http://www.ihtsdo.org/snomed-ct/

7.1.2 RPQs with Nesting

Recent works in the database field advocate the extension of RPQs with *nesting*, allowing one to require that objects along a path satisfy complex conditions, in turn expressed through (nested) 2RPQs, in line with the XML query language XPath. RPQs with nesting were proposed as the basic component of the navigational language nSPARQL for RDF (Pérez, Arenas, & Gutierrez, 2010) and they have received some attention in the setting of graph databases (Reutter, 2013; Barceló, Pérez, & Reutter, 2012).

Building on the conference version of this work, we recently studied the query answering problem for nested (C)2RPQs in the presence of DL ontologies (Bienvenu, Calvanese, Ortiz, & Šimkus, 2014). We establish tight complexity bounds in data and combined complexity for a variety of DLs, ranging from the lightweight DLs DL-Lite and $\mathcal{EL}$ considered in the present paper, up to highly expressive DLs. We show that adding nesting to (C)2RPQs does not increase the worst-case data complexity of query answering, but it leads to ExpTime-hardness in combined complexity, even for (non-conjunctive) 2RPQs and the lightweight DLs DL-Lite or $\mathcal{EL}$. This contrasts sharply with the tractability result we obtained in this paper for the same setting but without nesting.

Other authors have considered nested navigational queries over DL KBs. Stefanoni, Motik, Krötzsch, and Rudolph (2014) studied the complexity of answering certain types of nested path queries over knowledge bases formulated in OWL 2 EL, which extends $\mathcal{ELH}$ with a number of constructs, most notably *complex role inclusions*. They establish PSpace membership for a query language that roughly corresponds to the nested (C)RPQs mentioned earlier extended with unary complex labels, and they show P membership for the non-conjunctive fragment. These results demonstrate that nesting is computationally simpler when inverse roles are allowed neither in queries, nor in the ontology language. Kostylev, Reutter, and Vrgoc (2015) have recently investigated the complexity of the so-called DLXPath family of query languages over knowledge bases expressed in lightweight DLs, with a particular emphasis on the connection to propositional dynamic logic (PDL) and the effects of negation. The most expressive variant of DLXPath can be used to test Boolean conditions on nodes and edges (and thus fully captures the complex labels of the previous subsection), but unfortunately, answering such queries is undecidable even in the simplest of settings. Disallowing negation of binary relations restores decidability, but query answering remains coNP-hard in data complexity if negation of unary expressions is permitted. We finally note that Bourhis, Krötzsch, and Rudolph (2014) have recently explored several highly expressive extensions of RPQs with nesting for which query answering remains decidable in the presence of DL ontologies.

7.1.3 Path Variables and Path Relations

In the area of graph databases, it has been argued that C2RPQs are sometimes too weak since they can neither output the witnessing paths, nor talk about the relationships holding between different paths. This has motivated the recent extension of C2RPQs with *path variables* and *relations among tuples of paths* (Barceló et al., 2012; Barceló & Muñoz, 2014). The introduction of path variables makes it possible to refer to the specific paths (or more precisely, the labels of those paths) that are used to witness the satisfaction of query atoms. By using a path variable in multiple atoms, one can enforce that paths with the same label are used to connect different pairs of points. Moreover, path variables can appear as *answer variables* in the query, in which case a compact representation of the labels of witnessing paths is given as output (Barceló et al., 2012). A further extension allows queries to enforce that a tuple of path labels belongs to a given relation, either via *regular relations* such as prefix or equal length (Barceló et al., 2012), or using some common non-regular relations like subword and subsequence (Barceló & Muñoz, 2014).

An in-depth study of these extensions in the presence of DL ontologies is ongoing work, but our preliminary results suggest that the addition of ontological knowledge makes things significantly harder. For instance, even in the presence of very simple DL-Lite ontologies, the labels of paths witnessing a query answer may form a non-regular language, as illustrated by the following example.

Example 7.1. Consider the DL-Lite KB $\mathcal{K}$ consisting of the TBox $\{A \sqsubseteq \exists R, \exists R^- \sqsubseteq \exists R\}$ and ABox $\{A(a)\}$, and let q be the 2RPQ $\mathcal{E}(x, y)$ with $\mathcal{E} = r^* \cdot (r^-)^*$. There are infinitely many paths witnessing that (a, a) is an answer to q in $\mathcal{I}_{\mathcal{K}}$, each obtained by taking n steps away from a via r, then n steps back up to a via r^-. It follows that the set of labels of the witnessing paths forms the non-regular language $\{r^n \cdot (r^-)^n \mid n \geq 0\}$.

The previous example crucially uses inverse roles, but non-regular (indeed, non-context-free!) languages can also be enforced using shared path variables:

Example 7.2. Consider the $\mathcal{EL}$ KB $\mathcal{K}$ consisting of the TBox $\{A \sqsubseteq \exists r.A, A \sqsubseteq \exists s.A\}$ and ABox $\{A(a)\}$. Let q be the Boolean CRPQ $\exists x, y, z. \mathcal{E}(x, y) \wedge \mathcal{E}(y, z) \wedge \mathcal{E}(x, z)$ with $\mathcal{E} = (r \cup s)^*$, and further suppose that we use path variables to require that the label of the path from x to y is the same as the label of the path from y to z. Then every triple of paths (p_{xy}, p_{yz}, p_{xz}) witnessing the satisfaction of q is such that $\lambda(p_{xy}) = \lambda(p_{yz})$ and $\lambda(p_{xz}) = \lambda(p_{xy})\lambda(p_{yz})$. It follows that the set of labels of witnessing paths for the third atom yields the language $\{ww \mid w \in L((r \cup s)^*)\}$, which is neither regular nor context-free.

While these examples may seem artificial, they highlight the difficulties that arise when combining path variables and ontologies. In particular, they demonstrate that we cannot use NFAs as a compact representation of path labels (as was the case for graph databases), and even if all path variables are existentially quantified, the sharing of path variables will likely require significant modification of our query answering algorithms. Interestingly, it seems that some of the techniques based on word equations with regular constraints that were recently explored for handling non-regular path relations (Barceló & Muñoz, 2014) may be relevant in the presence of ontologies already for answering queries with (existentially quantified) path variables, and possibly also simple regular relations.

7.2 Other DLs

Our rewriting algorithm for C2RPQs is inspired by a technique first proposed for answering CQs in the DL Horn-$\mathcal{SHIQ}$ (Eiter et al., 2012), and it can be extended to all the constructs in that logic. In fact, a similar algorithm for nested C2RPQs was already developed for the DL $\mathcal{ELHI}^\perp$ which subsumes both DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$ (Bienvenu et al., 2014). $\mathcal{ELHI}^\perp$ contains many of the constructors of Horn-$\mathcal{SHIQ}$, and they behave similarly in terms of computational complexity. We point out that transitive roles, which are often problematic for query answering algorithms, are not a major issue in this setting. They can be easily accommodated in the $\mathcal{ELHI}^{\bot}$ algorithm by combining the known techniques for axiomatizing in the TBox the transitivity of roles, and the machinery for handling the transitive closure constructor in the queries. The extension of our algorithm to $\mathcal{ELHI}^{\bot}$ or Horn-$\mathcal{SHIQ}$ runs in polynomial time in the size of the data, and it is thus worst-case optimal in data complexity. Naturally, in combined complexity, it may require exponential time in some cases, but it always runs in single-exponential time, which again is worst-case optimal for these DLs. Moreover, we conjecture that if implemented smartly, this exponential behavior will rarely occur for real-world ontologies.

7.3 OWL 2 Profiles and Query Answering for the Semantic Web

The Web Ontology Language is a family of languages for specifying ontologies, endorsed as a standard by the W3C. The current version of the standard, called OWL 2 (OWL Working Group, 2009), features three *profiles* (Motik et al., 2012) or sublanguages that restrict the expressivity in such a way that logical inference over ontologies can be achieved by efficient algorithms. The three profiles provide different modeling capabilities, making them suitable for different applications: the EL profile is the preferred language for life science ontologies, the QL profile is geared towards applications that enrich relational data with ontological information, and the RL profile is used mostly for reasoning with Web data. For more information about the profiles, their modeling capabilities and supported inference services, we refer the reader to the introductory text by Krötzsch and references therein (Krötzsch, 2012). The $\mathcal{EL}$ and DL-Lite families of lightweight description logics studied in the present paper provide the logical underpinnings of the EL and QL profiles (the third profile, RL, is based upon Datalog). As a consequence, our results are immediately relevant to the problem of answering regular path queries over OWL 2 knowledge bases formulated using the QL and EL profiles. In particular, our algorithms can be adapted for querying datasets, such as RDF triplestores, enriched with ontological knowledge expressed in the fragments of the profiles that correspond to DL-Lite$_{\mathcal{R}}$ and $\mathcal{ELH}$.

8. Conclusion and Future Work

In this paper, we have provided algorithms and 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, for both DL-Lite$_{\mathcal{R}}$ and $\mathcal{ELH}$, query answering remains tractable in data complexity and in PSPACE in combined complexity for the highly expressive class of C2RPQs, and for 2RPQs, we even retain tractability in combined complexity. This computational price does not seem too high, particularly if we consider that the rich navigational features of these queries can partially compensate for the limited expressiveness of lightweight ontology languages. We thus believe that C2RPQs constitute a promising language for ontology-mediated query answering.

An important challenge for future work is to implement and experimentally evaluate the developed algorithms. Although C2RPQs is the natural query language to aim at, we believe that it makes more sense to start with a prototype implementation of the algorithm for (2)RPQs. Indeed, not only does the algorithm for (2)RPQs have a significantly lower worst-case complexity, but it is also considerably simpler than the rewriting-based approach for C2RPQs, and we are confident that it can be easily translated into a practical procedure. By contrast, the rewriting algorithm used to establish the PSpace upper bound for C2RPQs involves a considerable amount of non-determinism, and a naïve implementation can be expected to perform poorly. We nonetheless believe that the proposed rewriting approach, when suitably modified to avoid unnecessary non-deterministic guesses, provides a good basis for the development of practical methods for C2RPQ answering. Finally, identifying further restrictions on queries and ontologies that lead to lower combined complexity is another interesting problem for future study.

Acknowledgments

The authors would like to thank the anonymous reviewers for their very careful reading of the paper and their many helpful comments. This work was supported by the French National Research Agency (ANR) project PAGODA 12-JS02-007-01, the Austrian Science Fund (FWF) project T515, the FWF project P25518 and the Vienna Science and Technology Fund (WWTF) project ICT12-015.

Appendix A. Proof of Theorem 6.9

To complete the proof of Theorem 6.9, we must show how to handle queries of the form $\exists x.\, \alpha(x, x)$. As there does not seem to be a simple reduction to 2RPQs for queries of this form, we propose instead an approach based upon query rewriting.

To define our new query rewriting algorithm and prove its correctness, we will require the following notion. Given two concepts C, D from $\mathsf{BC}_{\mathcal{T}}$, we say that C *causes* D w.r.t. a TBox $\mathcal{T}$, denoted $C \rightsquigarrow_{\mathcal{T}} D$, if for every ABox $\mathcal{A}$ and any model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$, we have that $C^{\mathcal{I}} \neq \emptyset$ implies $D^{\mathcal{I}} \neq \emptyset$. It is not difficult to see that checking $C \rightsquigarrow_{\mathcal{T}} D$ is feasible in polynomial time:

Lemma A.1. *The following problem is in* P*: given a DL-Lite*$_{\mathcal{R}}$ *or* $\mathcal{ELH}$ *TBox* $\mathcal{T}$ *and concepts* $C, D \in \mathsf{BC}_{\mathcal{T}}$*, decide whether* $C \rightsquigarrow_{\mathcal{T}} D$*.*

Proof. We start with the case where $\mathcal{T}$ is a DL-Lite$_{\mathcal{R}}$ TBox. It is easy to see that $C \rightsquigarrow_{\mathcal{T}} D$ iff $\mathcal{T} \models C \sqsubseteq D$ or there exists a sequence of role names $R_1, \ldots, R_n$ such that

– $\mathcal{T} \models C \sqsubseteq \exists R_1$,
– $\mathcal{T} \models \exists R_n^- \sqsubseteq D$, and
– for $1 \leq i < n$, $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$.

We then observe that the existence of a sequence $R_1, \ldots, R_n$ satisfying these conditions can be decided in polynomial time by (i) initializing a set Reach with all roles R_1 such that $\mathcal{T} \models C \sqsubseteq \exists R_1$, (ii) saturating Reach by adding role S to Reach whenever $\mathcal{T} \models \exists U^- \sqsubseteq \exists S$ for some $U \in \mathsf{Reach}$, and (iii) checking whether there is some $U \in \mathsf{Reach}$ such that $\mathcal{T} \models \exists U^- \sqsubseteq D$. Since TBox reasoning is tractable for DL-Lite$_{\mathcal{R}}$, the procedure we have just described can be performed in polynomial time.

Assume $\mathcal{T}$ is an $\mathcal{ELH}$ TBox. We have that $C \rightsquigarrow_{\mathcal{T}} D$ iff $\mathcal{T} \models C \sqsubseteq D$ or there exists a sequence of concepts $\exists r_1.A_1, \ldots, \exists r_n.A_n$, where $\{A_1, \ldots, A_n\} \subseteq \mathsf{N_C}$, such that

- $\mathcal{T} \models C \sqsubseteq \exists r_1.A_1$,
- $\mathcal{T} \models A_n \sqsubseteq D$, and
- for $1 \leq i < n$, $\mathcal{T} \models A_i \sqsubseteq \exists r_{i+1}.A_{i+1}$.

The existence of a sequence $\exists r_1.A_1, \ldots, \exists r_n.A_n$ satisfying the above conditions can be decided in polynomial time, using a saturation procedure analogus to the one used for DL-Lite$_\mathcal{R}$ (recall that TBox reasoning is tractable for $\mathcal{ELH}$). □

In Figure 17, we present a deterministic query rewriting algorithm PolyRewrite that takes as input an NFA α and a DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ TBox $\mathcal{T}$ and outputs a set of queries, which will be denoted PolyRewrite$(\alpha, \mathcal{T})$. As was the case in Section 6, the purpose of query rewriting is to ensure that we only need to consider query matches that map answer variables to ABox individuals. Since we are only interested in queries of the form $\exists x. \alpha(x, x)$, it turns out that aside from the trivial rewriting $\alpha(x, x)$, it is sufficient to consider rewritings of the forms $C(x)$ or $C(x) \wedge \alpha_{s_1,s_2}(x, x)$ in which C is a basic concept and s_1 and s_2 are states in α. In Step 1, we initialize Frontier with all tuples (C, s_0, s_f) such that C is a basic concept, s_0 is the initial state of α, and s_f is a final state. Then, at each iteration of the while loop, we remove a tuple (C, s_1, s_2) from Frontier and add it to Visited to record that it has already been examined. If $C \in \mathsf{Loop}_\alpha[s_1, s_2]$, then the corresponding query $C(x) \wedge \alpha_{s_1,s_2}(x, x)$ is equivalent (under $\mathcal{T}$) to the simpler query $C(x)$, and so we add to Q all queries $D(x)$ that ensure that $\exists x. C(x)$ holds. If $C \notin \mathsf{Loop}_\alpha[s_1, s_2]$, then we add the corresponding query $C(x) \wedge \alpha_{s_1,s_2}(x, x)$ to Q. We next add to Frontier all those unvisited tuples (D, s_5, s_6) such that a match for $D(x) \wedge \alpha_{s_5,s_6}(x, x)$ that maps x to e implies the existence of a match for $C(x) \wedge \alpha_{s_1,s_2}(x, x)$ that maps x to a child of e in the anonymous part. This operation intuitively 'moves' the query match one step closer to the ABox and can be viewed as the analogue of Steps 6 and 7 in the algorithm Rewrite from Section 6.

A simple inspection of the algorithm PolyRewrite reveals that each tuple in $\mathsf{BC}_\mathcal{T} \times S \times S$ is examined at most once, and so there can be at most $|\mathsf{BC}_\mathcal{T}| \cdot |S|^2$ iterations of the while loop in Step 2. Since we know that all of the loop, entailment, and causation checks can be carried out in polynomial time, we obtain the following:

Lemma A.2. *The algorithm* PolyRewrite *runs in polynomial time in* $|\alpha|$ *and* $|\mathcal{T}|$*, and hence* PolyRewrite$(\alpha, \mathcal{T})$ *contains a polynomial number of queries.*

The next two lemmas establish the correctness of the rewriting procedure.

Lemma A.3. *If* $\mathcal{T}, \mathcal{A} \models \exists x. \alpha(x, x)$*, then there exists a query* $q(x) \in$ PolyRewrite$(\alpha, \mathcal{T})$ *which has a match* π *in* $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ *with* $\pi(x) \in \mathsf{Ind}(\mathcal{A})$.

Proof. Suppose that $\mathcal{T}, \mathcal{A} \models \exists x. \alpha(x, x)$, and let π be a match for $\exists x. \alpha(x, x)$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. If $\pi(x) \in \mathsf{Ind}(\mathcal{A})$, then the statement trivially holds since $\alpha(x, x)$ is added to Q in Step 1. Thus, suppose that $\pi(x) \notin \mathsf{Ind}(\mathcal{A})$. We start by proving the following claim, which captures how query matches in the anonymous part of the canonical model can be moved up closer to individuals:

Claim: Suppose that (C, s_1, s_2) is added to Frontier at some point during the execution of PolyRewrite on input $(\alpha, \mathcal{T})$. Further suppose that $C \notin \mathsf{Loop}_\alpha[s_1, s_2]$, and there is a match τ for $\alpha_{s_1,s_2}(x, x)$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\tau(x) = dRC$ for some $d \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Then there is a tuple

```
ALGORITHM PolyRewrite($\alpha, \mathcal{T}$)
Input: NFA $\alpha = (S, \Sigma, \delta, s_0, F)$, DL-Lite$_\mathcal{R}$ or $\mathcal{ELH}$ TBox $\mathcal{T}$
1. Set $Q = \{\alpha(x, x)\}$, $\mathsf{Visited} = \emptyset$, and $\mathsf{Frontier} = \{(C, s_0, s_f) \mid C \in \mathsf{BC}_\mathcal{T}, s_f \in F\}$.
2. While $\mathsf{Frontier} \neq \emptyset$
   (a) Move a tuple $(C, s_1, s_2)$ from Frontier to Visited[a].
   (b) If $C \in \mathsf{Loop}_\alpha[s_1, s_2]$, then
       • For every $D \in \mathsf{BC}_\mathcal{T}$ such that $D \rightsquigarrow_\mathcal{T} C$, add $D(x)$ to $Q$.
   (c) If $C \notin \mathsf{Loop}_\alpha[s_1, s_2]$, then
       • Add $C(x) \wedge \alpha_{s_1,s_2}(x, x)$ to $Q$.
       • For every tuple $(D, R, s_3, s_4, s_5, s_6) \in \mathsf{BC}_\mathcal{T} \times \mathsf{N}_\mathsf{R}^\pm \times S^4$ such that
           – $C \in \mathsf{Loop}_\alpha[s_1, s_3]$ and $C \in \mathsf{Loop}_\alpha[s_4, s_2]$,
           – $\mathcal{T} \models D \sqsubseteq \exists R$ and $C = \exists R^-$ [DL-Lite$_R$] or $\mathcal{T} \models D \sqsubseteq \exists R.C$ [$\mathcal{ELH}$ ],
           – there exist roles $R', R''$ with $\mathcal{T} \models R^- \sqsubseteq R'$, $\mathcal{T} \models R \sqsubseteq R''$, $(s_3, R', s_5) \in \delta$,
             and $(s_6, R'', s_4) \in \delta$, and
           – $(D, s_5, s_6) \notin (\mathsf{Frontier} \cup \mathsf{Visited})$,
         add $(D, s_5, s_6)$ to Frontier.
3. Output $Q$.
```

a. We choose the least tuple in Frontier according to some arbitrary lexicographic ordering on $\mathsf{BC}_\mathcal{T} \times S \times S$. We note however that the particular choice of a tuple does not affect the output of the procedure.

Figure 17: Query rewriting algorithm PolyRewrite.

(D, s_5, s_6) that is added to Frontier at some point such that the query $D(x) \wedge \alpha_{s_5,s_6}(x, x)$ has a match τ' such that $\tau'(x) = d$, $\mathcal{T} \models D \sqsubseteq \exists R.C$, and either $d \in \mathsf{Ind}(\mathcal{A})$ or $\mathsf{Tail}(d) = D$.

Proof of claim. Suppose that (C, s_1, s_2) and τ satisfy the conditions of the claim. Since $\tau(x) = dRC$, it follows from the definition of canonical models that there is a concept $D \in \mathsf{BC}_\mathcal{T}$ such that $d \in D^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and either $\mathcal{T} \models D \sqsubseteq \exists R$ and $C = \exists R^-$ (if $\mathcal{T}$ is formulated in DL-Lite$_R$), or $\mathcal{T} \models D \sqsubseteq \exists R.C$ (for $\mathcal{T}$ an $\mathcal{ELH}$ TBox). Moreover, we may choose D such that $D = \mathsf{Tail}(d)$ if $d \notin \mathsf{Ind}(\mathcal{A})$. We also know that τ is a match for $\alpha_{s_1,s_2}(x, x)$, so there exists a path $p = e_0u_1e_1u_2 \dots u_ne_n$ with $e_0 = e_n = \tau(x)$ whose label $\lambda(p)$ belongs to $L(\alpha_{s_1,s_2})$. Since $\lambda(p) \in L(\alpha_{s_1,s_2})$, we can find a sequence of states $s'_0s'_1 \dots s'_n$ with $s'_0 = s_1$ and $s'_n = s_2$ such that for every $1 \leq i \leq n$, $(s'_{i-1}, u_i, s'_i) \in \delta$. Because $C \notin \mathsf{Loop}_\alpha[s_1, s_2]$, we know that the match τ is not fully contained within $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_{\pi(x)}$. Thus, there must be at least one occurrence of the parent d of $\pi(x)$ in the path p. Let e_j and e_k respectively be the first and last occurrences of d in p (if there is a single occurrence of d, then $j = k$). Observe that $e_{j-1} = e_{k+1} = \pi(x)$. Set $s_3 = s'_{j-1}$, $s_4 = s'_{k+1}$, $s_5 = s'_j$, and $s_6 = s'_k$. The paths $e_0u_1e_1 \dots u_{j-1}e_{j-1}$ and $e_{k+1}u_{k+2}e_{k+2} \dots u_ne_n$ witness that $(\pi(x), \pi(x)) \in L(\alpha_{s_1,s_3})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and $(\pi(x), \pi(x)) \in L(\alpha_{s_4,s_2})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. As the paths $e_0 \dots e_{j-1}$ and $e_{k+1} \dots e_n$ begin and end at $\pi(x)$

and are fully contained within $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_{\pi(x)}$, we obtain $C \in \mathsf{Loop}_\alpha[s_1, s_3]$ and $C \in \mathsf{Loop}_\alpha[s_4, s_2]$. Finally, we know from the definiton of paths and the construction of the canonical model that there must exist roles R', R'' with $\mathcal{T} \models R^- \sqsubseteq R'$, $\mathcal{T} \models R \sqsubseteq R''$, $(s_3, R', s_5) \in \delta$, and $(s_6, R'', s_4) \in \delta$.

By assumption, the tuple (C, s_1, s_2) is added to $\mathsf{Frontier}$ at some point, and it will eventually be selected during Step 2. Moreover, since $C \notin \mathsf{Loop}_\alpha[s_1, s_2]$, we will enter 2(c) when examining (C, s_1, s_2). From what we have shown above, we know that the tuple $(D, R, s_3, s_4, s_5, s_6)$ satisfies the first three requirements of the for-loop in Step 2(c). If the fourth requirement also holds, then this means that (D, s_5, s_6) is added to $\mathsf{Frontier}$, and if it fails, then this is because this triple has already been added to $\mathsf{Frontier}$ earlier in the execution of the algorithm. To complete the proof of the claim, we remark that the path $e_j u_{j+1} \ldots u_k e_k$ witnesses that $(d, d) \in L(\alpha_{s_5,s_6})^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Moreover, we have seen that $d \in D^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. It follows that by setting $\tau'(x) = d$, we obtain a match for the query $D(x) \wedge \alpha_{s_5,s_6}(x, x)$ with the required properties. *(end proof of claim)*

Observe that if τ' as described in the claim exists, then $\tau'(x)$ is the parent of $\tau(x)$ in the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. We can now finalize the proof. Indeed, since $\pi(x) \notin \mathsf{Ind}(\mathcal{A})$, we have $\pi(x) = dRC$ for some $R \in \mathsf{N}_\mathsf{R}^\pm$ and $C \in \mathsf{BC}_\mathcal{T}$. As π is a match for $\alpha(x, x)$, it must also be a match for $\alpha_{s_0,s_f}(x, x)$ for some $s_f \in F$. In Step 1, the tuple (C, s_0, s_f) will be added to $\mathsf{Frontier}$. Repeated applications of the claim either yield a query $q(x) \in \mathsf{PolyRewrite}(\alpha, \mathcal{T})$ having a match τ mapping x to the ABox, or result in the insertion of a tuple (D, s_1, s_2) into $\mathsf{Frontier}$ such that $D(x) \wedge \alpha_{s_1,s_2}(x, x)$ has a match in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ and $D \in \mathsf{Loop}_\alpha[s_1, s_2]$. In the latter case, let τ be a match for $D(x) \wedge \alpha_{s_1,s_2}(x, x)$, and let $a \in \mathsf{Ind}(\mathcal{A})$ be such that $\tau(x) \in \mathcal{I}_{\mathcal{T},\mathcal{A}}|_a$. Then it follows from the definition of canonical models that there is some $E \in \mathsf{BC}_\mathcal{T}$ such that $E \rightsquigarrow_\mathcal{T} D$ and $a \in E^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. Thus, there is a match for $E(x) \in \mathsf{PolyRewrite}(\alpha, \mathcal{T})$ that maps x to $a \in \mathsf{Ind}(\mathcal{A})$. $\square$

Lemma A.4. *If $q(x) \in \mathsf{PolyRewrite}(\alpha, \mathcal{T})$ and π is a match for $q(x)$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ with $\pi(x) \in \mathsf{Ind}(\mathcal{A})$, then $\mathcal{T}, \mathcal{A} \models \exists x. \alpha(x, x)$.*

Proof. To simplify presentation we introduce the notion of containment of Boolean queries w.r.t. a TBox. Given a TBox $\mathcal{T}$ and two Boolean queries q_1, q_2, we write $q_1 \subseteq_\mathcal{T} q_2$ if for every ABox $\mathcal{A}$ we have that $\mathcal{T}, \mathcal{A} \models q_1$ implies $\mathcal{T}, \mathcal{A} \models q_2$. With a slight abuse of notation we allow q_1 and q_2 to contain atoms of the form $\exists R(x)$. Each occurrence of $\exists R(x)$ in a query q is a shorthand for $R(x, y)$ where y is a variable that occurs once in q.

We start by establishing the following claim:

Claim: If (C, s_1, s_2) is added to $\mathsf{Frontier}$ at some point during the execution of $\mathsf{PolyRewrite}$ on input $(\alpha, \mathcal{T})$, then $\exists x. C(x) \wedge \alpha_{s_1,s_2}(x, x) \subseteq_\mathcal{T} \exists x. \alpha(x, x)$.

Proof of claim. The proof is by induction on the precedence relation obtained by setting $(C, s, s') \prec (D, s'', s''')$ if the tuple (D, s'', s''') is added to $\mathsf{Frontier}$ during the examination of tuple (C, s, s'). For the base case, we have the tuples (C, s_0, s_f) which are inserted in Step 1. Every such tuple has the form (C, s_0, s_f), where $s_f \in F$, and thus we trivially have $\exists x. C(x) \wedge \alpha_{s_0,s_f}(x, x) \subseteq_\mathcal{T} \exists x. \alpha(x, x)$. Next suppose that we have already shown the property for (C, s_1, s_2), and let (D, s_5, s_6) be such that $(C, s_1, s_2) \prec (D, s_5, s_6)$. Further suppose that there is a match π for $D(x) \wedge \alpha_{s_5,s_6}(x, x)$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. Then we can find a path $p_0 = e_0 u_1 e_1 u_2 \ldots u_n e_n$ with $e_0 = e_n = \pi(x)$ such that $\lambda(p) \in L(\alpha_{s_5,s_6})$. It follows that there must exist a sequence of states $s'_0 s'_1 \dots s'_n$ with $s'_0 = s_5$ and $s'_n = s_6$ such that for every $1 \leq i \leq n$, $(s'_{i-1}, u_i, s'_i) \in \delta$. Because $(C, s_1, s_2) \prec (D, s_5, s_6)$, there must exist a tuple $(D, R, s_3, s_4, s_5, s_6)$ such that (D, s_5, s_6) was added to $\mathsf{Frontier}$ when examining $(D, R, s_3, s_4, s_5, s_6)$. We know that this tuple must have satisfied the four conditions, and so we must have

- $\mathcal{T} \models D \sqsubseteq \exists R$ and $C = \exists R^-$ [DL-Lite$_R$] or $\mathcal{T} \models D \sqsubseteq \exists R.C$ [$\mathcal{ELH}$],
- $C \in \mathsf{Loop}_\alpha[s_1, s_3]$ and $C \in \mathsf{Loop}_\alpha[s_4, s_2]$, and
- there exist roles R', R'' with $\mathcal{T} \models R^- \sqsubseteq R'$, $\mathcal{T} \models R \sqsubseteq R''$, $(s_3, R', s_5) \in \delta$, and $(s_6, R'', s_4) \in \delta$.

By the first point, the element $\pi(x)RC$ belongs to the canonical model, and by the second point, we can find paths $p_1 = e'_0 \dots u'_m e'_m$ and $p_2 = e''_0 \dots u''_\ell e''_\ell$ such that $e'_0 = e''_0 = e'_m = e''_\ell = \pi(x)RC$, $\lambda(p_1) \in L(\alpha_{s_1,s_3})$, and $\lambda(p_2) \in L(\alpha_{s_4,s_2})$. Using this and the third point, we can show that the path $p^* = p_1 R' p_0 R'' p_2$ is such that $\lambda(p^*) \in L(\alpha_{s_1,s_2})$. Since p^* begins and ends at $\pi(x)RC$ and $\pi(x)RC \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, it follows that $\mathcal{T}, \mathcal{A} \models \exists x.\, C(x) \wedge \alpha_{s_1,s_2}(x,x)$. By the induction hypothesis, $\exists x.\, C(x) \wedge \alpha_{s_1,s_2}(x,x) \subseteq_{\mathcal{T}} \exists x.\, \alpha(x,x)$, so we must also have $\mathcal{T}, \mathcal{A} \models \exists x.\, \alpha(x,x)$. This establishes the desired containment $\exists x.\, D(x) \wedge \alpha_{s_5,s_6}(x,x) \subseteq_{\mathcal{T}} \exists x.\, \alpha(x,x)$. (*end proof of claim*)

Now suppose that π is a match for $q(x) \in \mathsf{PolyRewrite}(\alpha, \mathcal{T})$ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi(x) \in \mathsf{Ind}(\mathcal{A})$. There are three possibilities. The first is that $q(x) = \alpha(x,x)$, in which case we trivially have $\mathcal{T}, \mathcal{A} \models \exists x.\, \alpha(x,x)$. The next possibility is that $q(x) = \exists x.\, C(x) \wedge \alpha_{s,s'}(x,x)$, in which case we can apply the preceding claim to show that $\mathcal{T}, \mathcal{A} \models \exists x.\, \alpha(x,x)$. The final possibility is that $q(x) = D(x)$, in which case there must have been some $(C, s, s') \in \mathsf{Frontier}$ such that $C \in \mathsf{Loop}_\alpha[s, s']$ and $\exists x.\, D(x) \subseteq_{\mathcal{T}} \exists x.\, C(x)$. In this case, we have $\exists x.\, D(x) \subseteq_{\mathcal{T}} \exists x.\, C(x)$, $\exists x.\, C(x) \subseteq_{\mathcal{T}} \exists x.\, C(x) \wedge \alpha_{s,s'}(x,x)$ (since $C \in \mathsf{Loop}_\alpha[s, s']$), and $\exists x.\, C(x) \wedge \alpha_{s,s'}(x,x) \subseteq_{\mathcal{T}} \exists x.\, \alpha(x,x)$ (by the claim). Putting these statements together, we obtain $\exists x.\, D(x) \subseteq_{\mathcal{T}} \exists x.\, \alpha(x,x)$, which yields $\mathcal{T}, \mathcal{A} \models \exists x.\, \alpha(x,x)$. $\square$

To complete the argument, we observe that the queries output by $\mathsf{PolyRewrite}$ are either 2RPQs or take the form $C(x)$ or $C(x) \wedge \alpha_{s,s'}(x,x)$, and queries of the latter forms are trivially transformed into 2RPQs. It follows that we can answer a query of the form $\exists x.\, \alpha(x,x)$ by (i) computing the set $\mathsf{PolyRewrite}(\alpha, \mathcal{T})$, and (ii) using $\mathsf{EvalAtom}$ to check, for each 2RPQ $\alpha'(x,x)$ obtained from $\mathsf{PolyRewrite}(\alpha, \mathcal{T})$, whether $\mathsf{cert}(\alpha'(x,x), (\mathcal{T}, \mathcal{A})) \neq \emptyset$. Correctness of this procedure follows from Proposition 5.8 and Lemmas A.3 and A.4, and the complexity bounds then follow from Lemma A.2 and Theorem 5.9.

References

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of Databases.* Addison-Wesley.

Arora, S., & Barak, B. (2009). *Computational Complexity - A Modern Approach.* Cambridge University Press.

Artale, A., Calvanese, D., Kontchakov, R., & Zakharyaschev, M. (2009). The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)*, *36*, 1–69.

Baader, F., Brandt, S., & Lutz, C. (2005). Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*.

Bala, S. (2002). Intersection of regular languages and star hierarchy. In *Proceedings of the Twenty-Ninth International Colloquium on Automata, Languages and Programming (ICALP 2002)*.

Barceló, P. (2013). Querying graph databases. In *Proceedings of the Thirty-Second Symposium on Principles of Database Systems (PODS 2013)*.

Barceló, P., Libkin, L., Lin, A. W., & Wood, P. T. (2012). Expressive languages for path queries over graph-structured data. *ACM Transactions on Database Systems (TODS)*, *37*(4), 31.

Barceló, P., & Muñoz, P. (2014). Graph logics with rational relations: The role of word combinatorics. In *Proceedings of the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2014)*.

Barceló, P., Pérez, J., & Reutter, J. L. (2012). Relative expressiveness of nested regular expressions. In *Proceedings of the Sixth Alberto Mendelzon International Workshop on Foundations of Data Management (AMW 2012)*.

Berglund, A., Boag, S., Chamberlin, D., Fernández, M. F., Kay, M., Robie, J., & Siméon, J. (2007). *XML Path Language (XPath) 2.0*. W3C Recommendation. Available at `http://www.w3.org/TR/xpath20/`.

Bienvenu, M., Calvanese, D., Ortiz, M., & Šimkus, M. (2014). Nested regular path queries in description logics. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2014)*.

Bienvenu, M., Ortiz, M., & Šimkus, M. (2012). Answering expressive path queries over lightweight DL knowledge bases. In *Proceedings of the Twenty-Fifth International Workshop on Description Logics (DL 2012)*.

Bienvenu, M., Ortiz, M., & Šimkus, M. (2013). Conjunctive regular path queries in lightweight description logics. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013)*.

Bourhis, P., Krötzsch, M., & Rudolph, S. (2014). How to best nest regular path queries. In *Proceedings of the Twenty-Seventh International Workshop on Description Logics (DL 2014)*.

Brickley, D., & Guha, R. (2014). *RDF Schema 1.1*. W3C Recommendation. Available at `http://www.w3.org/TR/rdf-schema/`.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2006). Data complexity of query answering in description logics. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*.

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

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1998). On the decidability of query containment under constraints. In *Proceedings of the Seventeenth Symposium on Principles of Database Systems (PODS 1998)*.

Calvanese, D., Eiter, T., & Ortiz, M. (2007). Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*.

Calvanese, D., Eiter, T., & Ortiz, M. (2009). Regular path queries in expressive description logics with nominals. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009)*.

Calvanese, D., Eiter, T., & Ortiz, M. (2014). Answering regular path queries in expressive description logics via alternating tree-automata. *Information and Computation*, *237*, 12–55.

Consens, M. P., & Mendelzon, A. O. (1990). GraphLog: A visual formalism for real life recursion. In *Proceedings of the Ninth Symposium on Principles of Database Systems (PODS 1990)*.

Ehrenfeucht, A., & Zeiger, P. (1974). Complexity measures for regular expressions. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing (STOC 1974)*.

Eiter, T., Ortiz, M., Šimkus, M., Tran, T., & Xiao, G. (2012). Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012)*.

Florescu, D., Levy, A., & Suciu, D. (1998). Query containment for conjunctive queries with regular expressions. In *Proceedings of the Seventeenth Symposium on Principles of Database Systems (PODS 1998)*.

Haase, C., & Lutz, C. (2008). Complexity of subsumption in the $\mathcal{EL}$ family of description logics: Acyclic and cyclic TBoxes. In *Proceedings of the Eighteenth European Conference on Artificial Intelligence (ECAI 2008)*.

Harris, S., & Seaborne, A. (2013). *SPARQL 1.1 Query Language*. W3C Recommendation. Available at `http://www.w3.org/TR/sparql11-query/`.

Kikot, S., Kontchakov, R., Podolskii, V. V., & Zakharyaschev, M. (2013). Query rewriting over shallow ontologies. In *Proceedings of the Twenty-Sixth International Workshop on Description Logics (DL 2013)*.

Kostylev, E. V., Reutter, J. L., & Vrgoc, D. (2015). XPath for DL ontologies. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*.

Kozen, D. (1977). Lower bounds for natural proof systems. In *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science (SFCS 1977)*.

Krisnadhi, A., & Lutz, C. (2007). Data complexity in the $\mathcal{EL}$ family of DLs. In *Proceedings of the Twentieth International Workshop on Description Logics (DL 2007)*.

Krötzsch, M. (2012). OWL 2 Profiles: An introduction to lightweight ontology languages. In *Proceedings of the Eighth Reasoning Web Summer School (RW 2012)*.

Krötzsch, M., & Rudolph, S. (2007). Conjunctive queries for $\mathcal{EL}$ with composition of roles. In *Proceedings of the Twentieth International Workshop on Description Logics (DL 2007)*.

Levy, A. Y., & Rousset, M. (1996). The limits on combining recursive Horn rules with description logics. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference (AAAI 96)*.

Lutz, C. (2008). The complexity of conjunctive query answering in expressive description logics. In *Proceedings of the Fourth Joint Conference on Automated Reasoning (IJCAR 2008)*.

Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., & Lutz, C. (2012). *OWL 2 Web Ontology Language Profiles*. W3C Recommendation. Available at `http://www.w3.org/TR/owl2-profiles/`.

Ortiz, M. (2013). Ontology based query answering: The story so far. In *Proceedings of the Seventh Alberto Mendelzon International Workshop on Foundations of Data Management (AMW 2013)*.

Ortiz, M., Rudolph, S., & Šimkus, M. (2011). Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011)*.

Ortiz, M., & Šimkus, M. (2012). Reasoning and query answering in description logics. In *Proceedings of the Eighth Reasoning Web Summer School (RW 2012)*.

Ortiz, M., & Šimkus, M. (2014). Revisiting the hardness of query answering in expressive description logics. In *Proceedings of the Eighth International Conference on Web Reasoning and Rule Systems (RR 2014)*.

OWL Working Group, W. (2009). *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation. Available at `http://www.w3.org/TR/owl2-overview/`.

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

Reutter, J. L. (2013). Containment of nested regular expressions. *CoRR*, *abs/1304.2637*.

Rosati, R. (2007). On conjunctive query answering in $\mathcal{EL}$. In *Proceedings of the Twentieth International Workshop on Description Logics (DL 2007)*.

Stefanoni, G., Motik, B., Krötzsch, M., & Rudolph, S. (2014). The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. *Journal of Artificial Intelligence Research (JAIR)*, *51*, 645–705.

Thompson, K. (1968). Regular expression search algorithm. *Communications of the ACM*, *11*(6), 419–422.