XPath for DL Ontologies

Egor V. Kostylev
University of Oxford

Juan L. Reutter
PUC Chile

Domagoj Vrgoč
PUC Chile

Abstract

Applications of description logics (DLs) such as OWL 2 and ontology-based data access (OBDA) require understanding of how to pose database queries over DL knowledge bases. While there have been many studies regarding traditional relational query formalisms such as conjunctive queries and their extensions, little attention has been paid to graph database queries, despite the fact that graph databases have essentially the same structure as DLs. In particular, not much is known about the interplay between DLs and XPath. The last is a powerful formalism for querying semistructured data: it is in the core of most practical query languages for XML trees, and it is also gaining popularity in theory and practice of graph databases. In this paper we make a step towards coupling knowledge bases and graph databases by studying how to answer powerful XPath-style queries over DL-Lite and EL. We start with adapting the definition of XPath to the DL context, and then proceed to study the complexity of evaluating XPath queries over knowledge bases. Results show that, while query answering is undecidable for the full XPath, by carefully tuning the amount of negation allowed in the queries we can arrive to XPath fragments that have a potential to be used in practical applications.

Introduction

Satisfiability and model checking have long been two central problems in the knowledge representation community. However new applications of description logics (DLs for short), such as the Web Ontology Language (OWL) (Motik et al. 2012) and ontology-based data access (OBDA), are forcing us to develop algorithms solving more complex data manipulation and extraction tasks, and in particular, require understanding how to answer database-style queries over knowledge bases that are specified by DLs (Glimm et al. 2013).

The literature on knowledge bases usually considers relational queries, mostly focusing on conjunctive queries (CQs) and their extensions with union (Calvanese et al. 2007; Artale et al. 2009), forms of negation (Rosati 2007; Gutiérrez-Basulto et al. 2013), and aggregates (Calvanese et al. 2008; Kostylev and Reutter 2013). Yet arguably the most natural database paradigm for querying DLs is graph databases, as these share the same structure with DL knowledge bases: they use unary and binary predicates (that is concepts and roles in DL terminology) to represent graph nodes and edges. While the idea of using graph queries in knowledge bases is not new (see e.g., (Calvanese et al. 2000)), we are only starting to understand how such queries can be fine tuned for various ontology languages. So far the specific research on this topic has primarily been concerned with the class of *regular path queries* (RPQs; see e.g. (Barceló 2013)), one of the most basic graph query languages. We now know how to evaluate such queries over various description logics (Calvanese, Eiter, and Ortiz 2007), how to deal with some of their extensions (Bienvenu, Ortiz, and Šimkus 2013; Bienvenu et al. 2014) and understand how to solve their containment problem in the presence of DL constraints (Calvanese, Ortiz, and Šimkus 2011).

There are of course many other languages for querying graph and semi-structured data, the most notable amongst them being XPath. Originally designed to extract data from XML trees (XPath 2.0 2010), XPath has recently been adapted to work over graph databases (Libkin, Martens, and Vrgoč 2013) and was shown to retain good evaluation properties while at the same time being more powerful than RPQs and many of their extensions. Moreover, XPath also subsumes navigational graph querying features of SPARQL 1.1 (Harris and Seaborne 2013), such as *property paths*, and other commercial graph query languages (see e.g., Neo4j (Robinson, Webber, and Eifrem 2013)). Therefore we can view XPath as a unifying formalism containing all of the usual querying primitives for graph data.

To get an impression of the type of queries one can ask in XPath consider the following: 'Can I fly from city A to city B making stops only in cities with UNESCO World Heritage Sites which are endangered?' If we assume that our ontology is modelled in a natural way (by having one binary predicate representing direct flights, another representing cities with UNESCO sites, and one unary predicate signifying if a site is endangered) this query cannot be expressed by RPQs, but can be expressed using XPath. Additionally, using XPath also allows us to reason about negated properties, such as requiring that none of the intermediate stops in the query above are listed under cultural criteria in the UNESCO classification—a feature not available in e.g. nested RPQs (Bienvenu et al. 2014).

Given that XPath is capable of expressing virtually all relevant querying primitives for graphs and is widely used by XML practitioners it is natural to ask whether it may have the same impact as a language for DL knowledge bases. But, as several studies in OBDA reveal, implementing standard database query languages over DLs is far from straightforward. Answering, for example, SQL queries over even the simplest of knowledge bases is known to be an undecidable task. In the same spirit, it is not unreasonable to think that the full XPath language might be too powerful to be used in DLs, and that first we need to find which fragments can be used efficiently (or indeed be used at all) in this context. In particular, XPath contains expressive primitives such as transitive closure and negation, and these are known to bring in conceptual difficulties even when studied in isolation.

The first step towards the acquisition of XPath technologies in the DL context is, then, to understand the interplay between XPath and DLs, pinpointing the features that might cause problems and identifying fragments for which their answers can be computed within reasonable complexity. To that extent, we first adapt the XPath query language to work over ontologies, arriving to DLXPath—an expressive language designed specifically for DL knowledge bases. To get a flavour of the interplay between DLXPath and DLs, we then study the problem of evaluating DLXPath queries over *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\text{core}}$ ontologies. Finally, we show how most of our techniques can then be applied to study query answering for ontologies belonging to the $\mathcal{EL}$ family. The choice for these particular families of DLs is twofold: they are simple enough to start such a research, but they are also quite important in practice, since they underly OWL 2 QL and OWL 2 EL profiles (Motik et al. 2012).

In this paper we distinguish two flavours of DLXPath: the *core* fragment, denoted by DLXPath$_{\text{core}}$, and the *regular* fragment, denoted by DLXPath$_{\text{reg}}$. The two are designed to match the duality present in the standard XPath query language (XPath 2.0 2010), and while the core fragment allows transitive closure only over basic role names, the regular fragment lifts this restriction, allowing us to post more general path queries. Regarding negation, we will distinguish full fragments, which allow both unary and binary negation, *path-positive* fragments with only the unary one, and *positive* fragments without any negation.

As usual when gauging the usefulness of a new query language for ontologies, we start by considering *data complexity* of the query answering problem, where one assumes that the query and the *terminological knowledge* (TBox) are fixed, while the only input is the *assertional knowledge* (ABox). Although we show that for the most general case the problem is undecidable, by limiting the amount of negation we obtain expressive languages whose queries can be answered in coNP and even NLogSpace, both over *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\text{core}}$. We then move to studying the *combined complexity*, where all of the knowledge base and query form the input. Here we obtain bounds ranging from NP-complete (thus matching the ones for ordinary CQs), to EXPTIME-complete, and undecidable for the full language. We would like to note that some of the results are obtained using the deep connection between DLs, DLXPath and *propositional dynamic logic*, thus providing some interesting new techniques for answering queries over ontologies. In the end we also show how our techniques can be extended to work over DLs of the $\mathcal{EL}$ family, including $\mathcal{ELHI}_{\bot}$, that is the DL subsuming both *DL-Lite* and plain $\mathcal{EL}$, thus providing an extensive overview of the behaviour of XPath based languages over lightweight DLs.

Due to the space restrictions, in the main version of this paper we only give ideas for the proofs of presented results. This is an extended version with an appendix containing full details of the proofs, submitted as supplemented material.

Preliminaries

The language of *DL-Lite*$_{\mathcal{R}}$ (and *DL-Lite*$_{\text{core}}$) (Calvanese et al. 2007; Artale et al. 2009) contains *individuals* $c_1, c_2, \ldots$, *concept names* $A_1, A_2, \ldots$, and *role names* $P_1, P_2, \ldots$. *Concepts* B and *roles* R are defined by the grammar

$$B \ ::= \ A_i \ \mid \ \exists R, \qquad\qquad R \ ::= \ P_j \ \mid \ P_j^-.$$

A *DL-Lite*$_{\mathcal{R}}$ *TBox* is a finite set of *concept* and *role inclusions* of the form

$$B_1 \sqsubseteq B_2, \quad B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqsubseteq R_2, \quad R_1 \sqcap R_2 \sqsubseteq \bot.$$

A *DL-Lite*$_{\text{core}}$ TBox contains only concept inclusions. An *ABox* is a finite set of *assertions* of the form $A_i(c_k)$ and $P_j(c_k, c_\ell)$. A *knowledge base* (*KB*) is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a nonempty domain $\Delta^{\mathcal{I}}$ of elements with an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns an element $c_k^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to each individual c_k, a subset $A_i^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ to each concept name A_i, and a binary relation $P_j^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role name P_j. When dealing with *DL-Lite* it is usual to adopt the *unique name assumption* (*UNA*), and we do so here by requiring that $c_k^{\mathcal{I}} \neq c_\ell^{\mathcal{I}}$, for all individuals $c_k \neq c_\ell$. Our results, however, do not depend on UNA. The interpretation function $\cdot^{\mathcal{I}}$ is extended to roles and concepts in the following standard way:

$$(\exists R)^{\mathcal{I}} \ = \ \big\{d \in \Delta^{\mathcal{I}} \mid \text{there is } d' \in \Delta^{\mathcal{I}} \text{ with } (d, d') \in R^{\mathcal{I}}\big\},$$
$$(P_j^-)^{\mathcal{I}} \ = \ \{(d', d) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d, d') \in P_j^{\mathcal{I}}\}.$$

The *satisfaction relation* $\models$ for TBox inclusions and ABox assertions is also standard:

$$\begin{array}{lll}
\mathcal{I} \models B_1 \sqsubseteq B_2 & \text{iff} & B_1^{\mathcal{I}} \subseteq B_2^{\mathcal{I}}, \\
\mathcal{I} \models B_1 \sqcap B_2 \sqsubseteq \bot & \text{iff} & B_1^{\mathcal{I}} \cap B_2^{\mathcal{I}} = \emptyset, \\
\mathcal{I} \models R_1 \sqsubseteq R_2 & \text{iff} & R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}, \\
\mathcal{I} \models R_1 \sqcap R_2 \sqsubseteq \bot & \text{iff} & R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} = \emptyset, \\
\mathcal{I} \models A_i(c_k) & \text{iff} & c_k^{\mathcal{I}} \in A_i^{\mathcal{I}}, \\
\mathcal{I} \models P_j(c_k, c_\ell) & \text{iff} & (c_k^{\mathcal{I}}, c_\ell^{\mathcal{I}}) \in P_j^{\mathcal{I}}.
\end{array}$$

A KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is *satisfiable* if there is an interpretation $\mathcal{I}$ satisfying all inclusions of $\mathcal{T}$ and assertions of $\mathcal{A}$. In this case we write $\mathcal{I} \models \mathcal{K}$ and say that $\mathcal{I}$ is a *model of* $\mathcal{K}$.

DLXPath: XPath for Knowledge Bases

As mentioned in the introduction, the family of *DL-Lite* was designed not only to keep satisfiability and model checking problems simple, but mainly to keep the complexity of conjunctive query answering the same as in the case of relational databases, while simultaneously maximising the expressive power of the ontological language. This allows to use *DL-Lite* as a foundation for practical data managing applications, such as OWL 2 QL and OBDA. However, this does not automatically mean that other useful query formalisms with good evaluation properties over databases also have good properties when posed over *DL-Lite* knowledge bases. Hence, each class of queries which can be useful in knowledge base applications requires a separate research on its computational properties.

In this paper we concentrate on an adaptation of XPath query language for XML trees to knowledge bases. Recently it was shown that (a version of) XPath can be successfully used for querying graph databases (Libkin, Martens, and Vrgoč 2013). Every interpretation of a *DL-Lite* vocabulary can be seen as a graph, and hence every *DL-Lite* KB is an incomplete description of a graph. That is why we expect our adaptation $\mathsf{DLXPath}$ for querying knowledge bases to be useful in practical applications.

In what follows, we will consider several fragments of $\mathsf{DLXPath}$. We start with $\mathsf{DLXPath}_{\text{core}}$, the fragment which corresponds to core XPath, the theoretical foundation of most practical languages for querying XML trees.[1]

Definition 1 Node formulas φ, ψ *and* path formulas α, β *of* $\mathsf{DLXPath}_{\text{core}}$ *are expressions satisfying the grammar*

$$\begin{array}{lcl} \varphi, \psi & := & A \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \langle \alpha \rangle, \\ \alpha, \beta & := & \varepsilon \mid R \mid [\varphi] \mid \alpha \cup \beta \mid \alpha \cdot \beta \mid \overline{\alpha} \mid R^+, \end{array} \tag{1}$$

where A ranges over concept names and R ranges over roles (i.e., role names and their inverses).

Semantics $[\![\cdot]\!]^{\mathcal{I}}$ of $\mathsf{DLXPath}_{\text{core}}$ for an interpretation $\mathcal{I}$ associates subsets of $\Delta^{\mathcal{I}}$ to node formulas and binary relations on $\Delta^{\mathcal{I}}$ to path formulas as given in Table 1.

As usual when dealing with ontologies, our interest is not query answering for a particular interpretation, but computing those answers that are true in all possible models of the knowledge base. Formally, let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and α a $\mathsf{DLXPath}$ path formula. The *certain answers* of α over $\mathcal{K}$, denoted $\mathit{Certain}(\alpha, \mathcal{K})$, is the set of all pairs (c_1, c_2) of individuals such that $(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \in [\![\alpha]\!]^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$. Similarly, one can define certain answers $\mathit{Certain}(\varphi, \mathcal{K})$ for a $\mathsf{DLXPath}$ node formula φ as the set of all individuals c such that $c^{\mathcal{I}} \in [\![\varphi]\!]^{\mathcal{I}}$, for all $\mathcal{I}$ models of $\mathcal{K}$. In the paper all of the results will be stated for path formulas (*queries* from here on), however, they remain unchanged for node formulas.

Example 2 Coming back to the example from the introduction, consider role names $\mathsf{HasDirectFlight}$, which connects cities with a direct flight, and $\mathsf{HasUNESCOSite}$, which connects cities with their UNESCO world heritage sites, as well as concept $\mathsf{InDanger}$ denoting that a particular site is endangered. Let KB $\mathcal{K}$ represents the flight destination graph, as well as knowledge about heritage sites, partially explicitly in the ABox, and partially implicitly, by means of TBox inclusions. Then checking whether it is possible to fly from Edinburgh to a city that has an endangered UNESCO world heritage site is equivalent to checking if *Edinburgh* is in $\mathit{Certain}(\varphi, \mathcal{K})$, where φ is a node formula

$$\langle \mathsf{HasDirectFlight}^+[\langle \mathsf{HasUNESCOSite}[\mathsf{InDanger}] \rangle] \rangle.$$

[1]The subscript 'core' is used in this paper for two unrelated purposes. This matching is historical and accidental, but we decided to stay with conventional notation despite this undesired collision.

$$\begin{array}{rcl} [\![A]\!]^{\mathcal{I}} & = & A^{\mathcal{I}} \\ [\![\neg\varphi]\!]^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus [\![\varphi]\!]^{\mathcal{I}} \\ [\![\varphi \wedge \psi]\!]^{\mathcal{I}} & = & [\![\varphi]\!]^{\mathcal{I}} \cap [\![\psi]\!]^{\mathcal{I}} \\ [\![\varphi \vee \psi]\!]^{\mathcal{I}} & = & [\![\varphi]\!]^{\mathcal{I}} \cup [\![\psi]\!]^{\mathcal{I}} \\ [\![\langle\alpha\rangle]\!]^{\mathcal{I}} & = & \{d \mid \text{there exists } d' \text{ such that } (d, d') \in [\![\alpha]\!]^{\mathcal{I}}\} \\ \\ [\![\varepsilon]\!]^{\mathcal{I}} & = & \{(d, d) \mid d \in \Delta^{\mathcal{I}}\} \\ [\![R]\!]^{\mathcal{I}} & = & R^{\mathcal{I}} \\ [\![[\varphi]]\!]^{\mathcal{I}} & = & \{(d, d) \mid d \in [\![\varphi]\!]^{\mathcal{I}}\} \\ [\![\alpha \cup \beta]\!]^{\mathcal{I}} & = & [\![\alpha]\!]^{\mathcal{I}} \cup [\![\beta]\!]^{\mathcal{I}} \\ [\![\alpha \cdot \beta]\!]^{\mathcal{I}} & = & [\![\alpha]\!]^{\mathcal{I}} \circ [\![\beta]\!]^{\mathcal{I}} \\ [\![\overline{\alpha}]\!]^{\mathcal{I}} & = & (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus [\![\alpha]\!]^{\mathcal{I}} \\ [\![R^+]\!]^{\mathcal{I}} & & \text{is the transitive closure of } R^{\mathcal{I}} \end{array}$$

Table 1: Semantics of $\mathsf{DLXPath}_{\text{reg}}$. The symbol '$\setminus$' stands for set-theoretic difference.

As already noted, the formalism of core XPath is in the nutshell of the most widespread query languages for XML trees. However, in (Libkin, Martens, and Vrgoč 2013) it was shown that the corresponding graph language cannot express certain properties that are deemed essential when querying graphs. Thus, besides $\mathsf{DLXPath}_{\text{core}}$ we also consider its generalisation called *regular* $\mathsf{DLXPath}$, or $\mathsf{DLXPath}_{\text{reg}}$, which extends the core fragment by allowing the use of transitive closure operator $^+$ over arbitrary path formulas. Formally, path formulas of $\mathsf{DLXPath}_{\text{reg}}$ satisfy the grammar

$$\alpha, \beta \;\; := \;\; \varepsilon \mid R \mid [\varphi] \mid \alpha \cup \beta \mid \alpha \cdot \beta \mid \overline{\alpha} \mid \alpha^+,$$

while node formulas remain the same as for $\mathsf{DLXPath}_{\text{core}}$ in grammar (1). As expected, the semantics $[\![\alpha^+]\!]^{\mathcal{I}}$ over an interpretation $\mathcal{I}$ is the transitive closure of $[\![\alpha]\!]^{\mathcal{I}}$.

Example 3 With full transitive closure we are now able to post more complex queries than in the core fragment. Consider again the knowledge base $\mathcal{K}$ from Example 2. We can now ask if it is possible to fly from Liverpool to Jerusalem, making stopovers only in places that have an endangered UNESCO world heritage site by checking if the pair $(\mathit{Liverpool}, \mathit{Jerusalem})$ is in $\mathit{Certain}(\alpha, \mathcal{K})$, where α is a query

$$(\mathsf{HasDirectFlight}[\langle \mathsf{HasUNESCOSite}[\mathsf{InDanger}] \rangle])^+.$$

Note that here we check if each of the cities along the path has an endangered site. If we want to additionally require that the sites along the route are not included in the UNESCO list under cultural criteria, we need to replace $[\mathsf{InDanger}]$ with $[\mathsf{InDanger} \wedge \neg\mathsf{Cultural}]$ in the query above.

Besides these two query languages, we will consider their fragments, which will be introduced as needed.

Before passing on to the complexity of DLXPath query evaluation, we briefly compare our languages with other formalisms. First, DLXPath is clearly incomparable with CQs. However, all tree-shaped CQs and unions of CQs with no more than two free variables can be written as $\mathsf{DLXPath}_{\text{core}}$ queries. On the other hand, all negation-free and $^+$-free $\mathsf{DLXPath}_{\text{reg}}$ queries can be written as unions of CQs, though with a cost of exponential blow-up. If we allow negation but only over concept and role names, then a query can be written as a union of CQs with safe negation. Second, $\mathsf{DLXPath}_{\text{reg}}$ queries without negation and node tests $[\varphi]$ are *2-way regular path queries* (*2RPQs*), the standard formalism for querying graph databases. If, additionally, role inverses are not allowed as path formulas, it becomes plain *RPQs*, that is, essentially, regular expressions. Some fragments of DLXPath can be expressed in other description logic formalisms. For example, it is well-known that a unary tree-shaped CQ with the corresponding tree being rooted and directed, can be written as a concept of $\mathcal{EL}$, a DL underlying OWL 2 EL profile (Motik et al. 2012). Finally, $\mathsf{DLXPath}_{\text{reg}}$ node formulas without path negation are nothing else but *propositional dynamic logic with converse* (*CPDL*) formulas; we will discuss this connection in more detail later on.

In the following sections we analyse the complexity of evaluating DLXPath queries over DL-Lite knowledge bases.

Data Complexity of DLXPath Query Evaluation

As it is widely accepted in theory and proved in practice, the size of the query and TBox is usually much smaller than the size of the ABox (see e.g., (Vardi 1982) for discussion in the relational database context and (Calvanese et al. 2007) for DLs). This is why one usually considers *data complexity* of query answering, assuming that the TBox and the query are fixed, and only ABox is part of the input. In this section we study this problem for various fragments of DLXPath. Formally, let $\mathcal{T}$ be a TBox and α a DLXPath query. We are interested in the following family of problems.

CERTAIN ANSWERS $(\alpha, \mathcal{T})$
Input: ABox $\mathcal{A}$ and pair (c_1, c_2) of individuals
Question: Is $(c_1, c_2) \in \mathit{Certain}(\alpha, (\mathcal{T}, \mathcal{A}))$?

As it was previously mentioned, data complexity of CQ query answering over *DL-Lite*$_\mathcal{R}$ knowledge bases is the same as over relational databases, that is, in LOGSPACE (Calvanese et al. 2007). At the first glance, one may expect a similar result in the case of DLXPath, where the complexity is NLOGSPACE-complete over graph databases (Libkin, Martens, and Vrgoč 2013). However, the combination of the open world assumption and allowing negation in queries makes things quite different. In fact, the situation here is more like in the case of CQ with safe negation, where the complexity jump is dramatic: from polynomiality to undecidability (Gutiérrez-Basulto et al. 2013).

Theorem 4 *There exists a DL-Lite*$_{\text{core}}$ *TBox* $\mathcal{T}$ *and a* $\mathsf{DLXPath}_{\text{core}}$ *query* α *such that the problem* CERTAIN ANSWERS $(\alpha, \mathcal{T})$ *is undecidable.*

The proof uses similar techniques as the proof of Theorem 1 in (Gutiérrez-Basulto et al. 2013), that shows the undecidability of the problem of computing certain answers for conjunctive queries with safe negation over *DL-Lite*$_\mathcal{R}$ knowledge bases. In fact, the reduction can be done using the empty TBox, which shows that undecidability is 'contained' in the formulation of the fixed query.

Having this negative result, a natural direction is to search for DLXPath fragments that have decidable certain answers problem. Based on previous studies for XPath in XML and graph databases (see e.g., (Benedikt, Fan, and Geerts 2008)), the most problematic primitive seems to be the negation $\bar{\alpha}$ in path formulas. Indeed, we now show that removing this primitive from the syntax leads to decidability of the certain answers problem. We write $\mathsf{DLXPath}^{\text{path-pos}}_{\text{core}}$ for the fragment of $\mathsf{DLXPath}_{\text{core}}$ that does not allow the negation $\bar{\alpha}$, for α a path formula, and define the fragment $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ accordingly. We then have the following theorem.

Theorem 5 *There exists a DL-Lite*$_{\text{core}}$ *TBox* $\mathcal{T}$ *and a* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{core}}$ *query* α *such that the problem* CERTAIN ANSWERS $(\alpha, \mathcal{T})$ *is* CONP*-hard. The problem is in* CONP *for any DL-Lite*$_\mathcal{R}$ *TBox* $\mathcal{T}$ *and* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ *query* α.

Similarly to the previous result, the reduction for hardness can be done using the empty TBox. Moreover, the fixed query does not use the transitive closure operator $^+$.

While this theorem looks positive in light of the general undecidability result, the complexity might still be too high for practical applications, as it leads to algorithms which run in exponential time in the size of the ABox. To lower the complexity and obtain tractable algorithms, one could also consider fragments of DLXPath that do not allow any form of negation, neither in node nor in path formulas. We denote such fragments of $\mathsf{DLXPath}_{\text{core}}$ and $\mathsf{DLXPath}_{\text{reg}}$ by $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ and $\mathsf{DLXPath}^{\text{pos}}_{\text{reg}}$, respectively. This last fragment is nothing else but *nested 2RPQs* (Pérez, Arenas, and Gutierrez 2010), a language that has already been studied for *DL-Lite* knowledge bases (Bienvenu et al. 2014), where an NLOGSPACE tight complexity bound was shown for the problem of computing certain answers for both *DL-Lite*$_{\text{core}}$ and *DL-Lite*$_\mathcal{R}$. Furthermore, by adapting known results from graph databases, it is not difficult to extend this result to show that the problem is already NLOGSPACE hard even for $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ queries (see e.g., (Barceló 2013) for a survey on such techniques). From these results we conclude that nesting and inverse in regular expressions, as well as fixed *DL-Lite*$_\mathcal{R}$ TBoxes do not increase the tractable data complexity of query evaluation.

Combined Complexity of DLXPath Query Evaluation

Even if data complexity is the most important measure in practice, *combined complexity*, that is complexity under the assumption that both TBox, ABox and the query are given as input, allows us to get a better understanding of the query answering problem, and often provides a blueprint of how to solve the problem in practice. That is why we continue our study in this direction. Formally, we consider the following family of problems, where X ranges over $\{core, \mathcal{R}\}$, y over $\{\text{core}, \text{reg}\}$, and z is either nothing, or 'path-pos', or 'pos'.

$\mathsf{DLXPath}^z_y$ CERTAIN ANSWERS OVER *DL-Lite*$_X$	
Input:	*DL-Lite*$_X$ KB $\mathcal{K}$, $\mathsf{DLXPath}^z_y$ query α, and pair (c_1, c_2) of individuals
Question:	Is $(c_1, c_2) \in \mathcal{C}ertain(\alpha, \mathcal{K})$?

As it immediately follows from Theorem 4, the problem remains undecidable for full $\mathsf{DLXPath}_{\text{core}}$ and, hence, for full $\mathsf{DLXPath}_{\text{reg}}$. However, the last result also follows from the connection of DLXPath with propositional dynamic logic (PDL) and some well known properties of PDL. Since this connection will be heavily used in the remainder of the paper we now discuss it in more detail.

First of all, we note that in PDL community a different terminology is used: for example, interpretations are called *Kripke structures*, concept names are called *propositional letters* or *variables*, role names are *atomic programs*, and inverse operator is *converse*. Though, to be consistent with the rest of the paper we stay with the DL terminology.

As already mentioned, node formulas of $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ are, essentially, PDL with converse (CPDL) formulas (see (Harel, Kozen, and Tiuryn 2000) for a good introduction on the topic). Formally, the syntax of plain *propositional dynamic logic* (*PDL*) is the same as $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ (i.e., CPDL), except that it does not allow the inverse P_i^- of role names as path expressions. The standard problem in the PDL community is the *satisfiability* of a node formula φ; that is, checking whether there exists an interpretation $\mathcal{I}$ and an element d in its domain such that φ holds in d.

Lutz et al. showed that the satisfiability of PDL formulas extended with arbitrary path negation (such a logic is denoted *PDL*$^\neg$) is undecidable (Lutz and Walther 2005), which already implies the undecidability of the certain answers problem $\mathsf{DLXPath}_{\text{reg}}$: indeed, a $\mathsf{PDL}^\neg$ formula φ is satisfiable if and only if the $\mathsf{DLXPath}_{\text{reg}}$ query $[\neg\varphi]$ has the certain answer (c, c) over the empty KB with individual c in the language. Note, however, that it does not immediately imply undecidability in data complexity shown in Theorem 4.

Turning our attention to the certain answers problem for $\mathsf{DLXPath}^{\text{path-pos}}$ queries, we again use the connection with the theory of PDL. It is well-known that the satisfiability problem for PDL is EXPTIME-complete (Harel, Kozen, and Tiuryn 2000). It remains in EXPTIME even if these formulas are allowed to use the transitive closure operator $^+$ only over role names. The same holds for CPDL (Harel, Kozen, and Tiuryn 2000) and *PDL*$^{(\neg)}$, that is, the extension of PDL which allows path negation in a limited form—only on role names (Lutz and Walther 2005). Similarly to the previous undecidability result, the EXPTIME lower bounds for these classes of PDL formulas already imply EXPTIME-hardness of the certain answers problem for $\mathsf{DLXPath}^{\text{path-pos}}_{\text{core}}$, even for empty KBs. Also, in (Bienvenu et al. 2014) hardness was established even for $\mathsf{DLXPath}^{\text{pos}}_{\text{reg}}$.

The upper bound is, however, much more challenging. To deal with *DL-Lite*$_\mathcal{R}$ knowledge bases, in particular with role inclusions, we join the aforementioned extensions of PDL with inverse and negation on role names, and consider the language *CPDL*$^{(\neg)}$ whose node formulas obey the same grammar (1) as PDL (and DLXPath), and path formulas are defined as follows:

$$\alpha, \beta \quad := \quad \varepsilon \mid R \mid [\varphi] \mid \alpha \cup \beta \mid \alpha \cdot \beta \mid \overline{R} \mid \alpha^+.$$

We need the following result to establish complexity of the certain answers problem.

Theorem 6 *Checking satisfiability of a* $\mathsf{CPDL}^{(\neg)}$ *node formula can be done in* EXPTIME.

The proof makes use of ideas from (Lutz and Walther 2005) and (Vardi and Wolper 1986). Although it is not strictly related to description logics and knowledge representation, we state the result explicitly as we believe it might be of interest to the PDL community.

Of course, satisfiability results do not transfer directly to query answering over KBs. However, widening and recasting the ideas from (De Giacomo and Lenzerini 1994) and (De Giacomo and Lenzerini 1996), we obtain the desired upper bound.

Lemma 7 *The problem* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ CERTAIN ANSWERS OVER *DL-Lite*$_\mathcal{R}$ *is in* EXPTIME.

Summing up, we obtain the following theorem (the hardness results follows from (Harel, Kozen, and Tiuryn 2000) and (Bienvenu et al. 2014) and are included just for completeness).

Theorem 8 *The problem* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ CERTAIN ANSWERS OVER *DL-Lite*$_\mathcal{R}$ *is* EXPTIME*-complete. It remains* EXPTIME*-hard for* $\mathsf{DLXPath}^{\text{pos}}_{\text{reg}}$ *queries with DL-Lite*$_{\text{core}}$ *KBs, and for* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{core}}$ *even with empty KBs.*

The only remaining fragment is that of $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ queries, that is, the restriction of the $\mathsf{DLXPath}_{\text{core}}$ language that use neither binary nor unary negation. In the previous section we saw that data complexity of answering $\mathsf{DLXPath}^{\text{pos}}$ queries is the same as answering RPQs and 2RPQs, regardless of whether we made use of the regular or the core fragment. For combined complexity, the case is now different. We have already seen that query answering remains EXPTIME-hard for $\mathsf{DLXPath}^{\text{pos}}$. We now show that the restriction to the core fragment decreases the complexity by almost one exponential.

Theorem 9 *The problem* $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ CERTAIN ANSWERS OVER *DL-Lite*$_\mathcal{R}$ *is* NP*-complete. It remains* NP*-hard for DL-Lite*$_{\text{core}}$.

It is worth to mention, that the result holds even if the queries are not allowed to use the transitive closure operator $^+$ at all, being, essentially, very restricted form of unions of CQs but with the same complexity of query answering. Hence, the border between tractable and intractable combined complexity of the problem lies somewhere very close to here, leaving 2RPQs on one side and $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ with CQs on the other.

	$\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$		$\mathsf{DLXPath}^{\text{pos}}_{\text{reg}}$		$\mathsf{DLXPath}^{\text{path-pos}}$		$\mathsf{DLXPath}$	
	data	combined	data	combined	data	combined	data	combined
DL-Lite	NLOGSPACE-c	**NP-c**	NLOGSPACE-c	EXPTIME-c	**coNP-c**	**EXPTIME-c**	**undec.**	**undec.**
$\mathcal{EL}$	PTIME-c	**EXPTIME-c*/NP-c†**	PTIME-c	EXPTIME-c	**coNP-c**	**EXPTIME-c**	**undec.**	**undec.**

Table 2: A summary of the complexity results. Here "-c" stands for "-complete" and "undec." for "undecidable". All the results hold for both $\mathsf{DLXPath}_{\text{core}}$ and $\mathsf{DLXPath}_{\text{reg}}$ unless a subscript core or reg has been added. The results in the first line hold for both *DL-Lite*$_{\text{core}}$ and *DL-Lite*$_{\mathcal{R}}$. The results in the second line hold for all of $\mathcal{EL}$, $\mathcal{ELH}_\bot$ and $\mathcal{ELHI}_\bot$, except for † that holds up to $\mathcal{ELH}_\bot$. The EXPTIME bound in * is shown for $\mathcal{ELHI}_\bot$. The new results of this paper are set off in bold.

More Expressive Logics

Another important family of lightweight DLs used in practice is $\mathcal{EL}$ and its extensions, which underlie the OWL 2 EL profile (Motik et al. 2012). Here we look at one particular logic from this family, denoted $\mathcal{ELHI}_\bot$, that is essentially the minimal DL from the $\mathcal{EL}$ family that that subsumes *DL-Lite*$_{\mathcal{R}}$. The increase in expressive power that comes with $\mathcal{ELHI}_\bot$ does come with an exponential jump in answering, for example, standard conjunctive queries. Surprisingly, the situation is different for $\mathsf{DLXPath}$ queries and almost all the results from the previous sections hold for all the range of DLs from $\mathcal{EL}$ to $\mathcal{ELHI}_\bot$. It is interesting to note that for these results we use the same base techniques outlined in the previous sections, which suggests that the techniques introduced in this paper are robust to the particular choice of description logics.

Formally, the language of $\mathcal{ELHI}_\bot$ (Baader, Brandt, and Lutz 2005; 2008) has the same roles as *DL-Lite*$_{\mathcal{R}}$, but allows for complex concepts of the form:

$$C \;::=\; \top \mid A_i \mid \exists R.C \mid C_1 \sqcap C_2,$$

where A_i is a concept name and R a role. The interpretation $\mathcal{I}$ is defined similarly as in the case of *DL-Lite*, with $\top^{\mathcal{I}}$ denoting the universal relation, $\bot^{\mathcal{I}}$ the empty set and $(C_1 \sqcap C_2)^{\mathcal{I}}$ the intersection of $C_1^{\mathcal{I}}$ and $C_2^{\mathcal{I}}$. Finally we define

$$(\exists R.C)^{\mathcal{I}} = \left\{d \in \Delta^{\mathcal{I}} \mid \exists d' \in \Delta^{\mathcal{I}} \colon (d,d') \in R^{\mathcal{I}}, d' \in C^{\mathcal{I}}\right\}.$$

An $\mathcal{ELHI}_\bot$ TBox is a set of concept inclusions of the form

$$C_1 \sqsubseteq C_2, \qquad C \sqsubseteq \bot$$

and role inclusions as in *DL-Lite*$_{\mathcal{R}}$. Plain $\mathcal{EL}$ allows only for concept inclusions without $\bot$ and role inverses, and $\mathcal{ELH}_\bot$ disallows role inverses.

When studying the complexity of $\mathsf{DLXPath}$ query answering over $\mathcal{ELHI}_\bot$ we inherit all of the lower bound from the results on *DL-Lite*, in particular we immediately obtain that query answering is undecidable for the full language. By carefully examining the proof of Theorem 4 we can also observe that the result there also holds for plain $\mathcal{EL}$.

What is more interesting is the fact that the EXPTIME algorithm for combined complexity of $\mathsf{DLXPath}^{\text{path-pos}}$ is robust enough to extend to this setting. Complementing this observation with the results for $\mathsf{DLXPath}^{\text{pos}}_{\text{reg}}$ from (Bienvenu et al. 2014), we can obtain the following theorem.

Theorem 10 *Answering* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ *queries over* $\mathcal{ELHI}_\bot$ *KBs is* EXPTIME-*complete in combined complexity, with the lower bound already holding for* $\mathcal{EL}$ *and* $\mathsf{DLXPath}^{\text{pos}}_{\text{reg}}$*. In data complexity the problem is* coNP-*complete for* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ *and drops to* PTIME-*complete for* $\mathsf{DLXPath}^{\text{pos}}_{\text{reg}}$*, again for both* $\mathcal{ELHI}_\bot$ *and* $\mathcal{EL}$*.*

The last remaining fragment we study is $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$. Here we show that the rise in data complexity is similar to the one in the previous theorem, that is, from NLOGSPACE-complete to PTIME-complete. This can be attributed to the expressive power of logics in the $\mathcal{EL}$ family, since hardness results already hold for instance queries over simple $\mathcal{EL}$ knowledge bases (Bienvenu, Ortiz, and Šimkus 2013). The NP bound for combined complexity that we showed for *DL-Lite*$_{\mathcal{R}}$ holds for $\mathcal{ELH}_\bot$ as well, but for full $\mathcal{ELHI}_\bot$ the problem becomes EXPTIME-complete.

Theorem 11 *Answering* $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ *queries over* $\mathcal{ELH}_\bot$ *KBs is* NP-*complete in combined complexity. For* $\mathcal{ELHI}_\bot$ *KBs it becomes* EXPTIME-*complete. For both DLs the problem is* PTIME-*complete in data complexity. Apart from combined complexity for* $\mathcal{ELHI}_\bot$ *KBs the lower bounds already hold for* $\mathcal{EL}$*.*

Conclusions and Future Work

In this paper we conducted a detailed study about using the XPath language to query ontologies of the DL-Lite and $\mathcal{EL}$ families. The results, summarised in Table 2, show that although the problem is generally undecidable, by limiting the amount of negation we can get decidable and even tractable fragments. The deep connection between XPath, DL and PDL allowed us to use ideas developed in other areas and gave insight on how query evaluation can be affected by certain aspects of the language. Although this connection was explored previously (see e.g., discussion in Chapters 13 and 14 in (Blackburn, Benthem, and Wolter 2006)), we believe that the time is ripe for a comprehensive survey describing how techniques from one area can be transferred to another and hope that the results of this paper can motivate such a survey.

From a practical point of view, we would like to find and test the classes of queries that are of particular practical interest and have either tractable general algorithms or reliable heuristics. Here we primarily want to tackle positive and path-positive fragments of XPath, as these queries were implemented and tested by the XML community, and many good heuristics have been developed over the years.

References

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The DL-Lite family and relations. *J. Artif. Intell. Res. (JAIR)* 36:1–69.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the EL envelope. In Kaelbling, L. P., and Saffiotti, A., eds., *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, 364–369. Professional Book Center.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the el envelope further. In Clark, K., and Patel-Schneider, P. F., eds., *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*.

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

Benedikt, M.; Fan, W.; and Geerts, F. 2008. XPath satisfiability in the presence of DTDs. *J. ACM* 55(2).

Bienvenu, M.; Calvanese, D.; Ortiz, M.; and Šimkus, M. 2014. Nested regular path queries in description logics. *In KR*.

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

Blackburn, P.; Benthem, J. F. A. K. v.; and Wolter, F. 2006. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. New York, NY, USA: Elsevier Science Inc.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. 2000. Containment of conjunctive regular path queries with inverse. In *7th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 176–185.

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

Calvanese, D.; Kharlamov, E.; Nutt, W.; and Thorne, C. 2008. Aggregate queries over ontologies. In Elmasri, R.; Doerr, M.; Brochhausen, M.; and Han, H., eds., *ONISW*, 97–104. ACM.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2007. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *AAAI*, 391–396.

Calvanese, D.; Ortiz, M.; and Šimkus, M. 2011. Containment of regular path queries under description logic constraints. In *IJCAI*, 805–812.

De Giacomo, G., and Lenzerini, M. 1994. Boosting the Correspondence between Description Logics and Propositional Dynamic Logics. In *AAAI*, 205–212.

De Giacomo, G., and Lenzerini, M. 1996. TBox and ABox Reasoning in Expressive Description Logics. In *KR*, 316–327.

Glimm, B.; Ogbuji, C.; Hawke, S.; Herman, I.; Parsia, B.; Polleres, A.; and Seaborne, A. 2013. SPARQL 1.1 entailment regimes. W3C Recommendation 21 March 2013, `http://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/`.

Gutiérrez-Basulto, V.; Ibáñez-García, Y. A.; Kontchakov, R.; and Kostylev, E. V. 2013. Conjunctive queries with negation over dl-lite: A closer look. In *RR*, 109–122.

Harel, D.; Kozen, D.; and Tiuryn, J. 2000. *Dynamic Logic*. MIT Press.

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

Kostylev, E. V., and Reutter, J. L. 2013. Answering counting aggregate queries over ontologies of the DL-Lite family. In *AAAI*.

Libkin, L.; Martens, W.; and Vrgoč, D. 2013. Querying graph databases with XPath. In *ICDT*, 129–140.

Lutz, C., and Walther, D. 2005. PDL with Negation of Atomic Programs. *Journal of Applied Non-Classical Logics* 15(2):189–213.

Motik, B.; Cuenca Grau, B.; Horrocks, I.; Wu, Z.; Fokoue, A.; and Lutz, C. 2012. OWL 2 Web Ontology Language Profiles (2nd Edition). W3C Recommendation.

Pérez, J.; Arenas, M.; and Gutierrez, C. 2010. nSPARQL: A navigational language for RDF. *Web Semantics: Science, Services and Agents on the World Wide Web* 8(4):255 – 270.

Robinson, I.; Webber, J.; and Eifrem, E. 2013. *Graph databases*. " O'Reilly Media, Inc.".

Rosati, R. 2007. The limits of querying ontologies. In *Proc. of the 11th Int. Conf. on Database Theory (ICDT)*, volume 4353 of *LNCS*, 164–178. Springer.

Stefanoni, G.; Motik, B.; and Horrocks, I. 2013. Introducing nominals to the combined query answering approaches for el. In *AAAI*.

Vardi, M. Y., and Wolper, P. 1986. Automata-Theoretic Techniques for Modal Logics of Programs. *J. Comput. Syst. Sci.* 32(2):183–221.

Vardi, M. Y. 1982. The complexity of relational query languages (extended abstract). In *STOC*, 137–146.

2010. XML Path Language (XPath) 2.0 (Second Edition). www.w3.org/TR/xpath20.

Appendix

In this appendix we give the proofs for all theorems of the paper. For presentational purposes of the paper, some of the proofs refer to constructions from other proofs described after, but of course there are no circular references.

Theorem 4 *There exists a DL-Lite*$_{\text{core}}$ *TBox* $\mathcal{T}$ *and a* $\mathsf{DLXPath}_{\text{core}}$ *query* α *such that the problem* CERTAIN ANSWERS $(\alpha, \mathcal{T})$ *is undecidable.*

Proof. In this proof we use intersection operator $\cap$ for $\mathsf{DLXPath}$ path formulas. This does not increase the expressive power of the language, since the intersection $\alpha_1 \cap \alpha_2$ can be written as $\overline{\bar{\alpha}_1 \cup \bar{\alpha}_2}$.

The proof is by reduction of the halting problem for deterministic Turing machines. In particular, given a Turing machine M, we construct a $\mathsf{DLXPath}_{\text{core}}$ query α such that M does not accept an input $\vec{w}$ encoded as an ABox $\mathcal{A}_{\vec{w}}$ and an element c_1 iff $(c_1, c_1) \notin \mathcal{C}ertain(\alpha, (\emptyset, \mathcal{A}_{\vec{w}}))$. (Note that $\boldsymbol{q}$ depend on M but not on $\vec{w}$, and we use only empty TBox). Applying this construction to a fixed deterministic *universal* Turing machine, that is, a machine that accepts its input $\vec{w}$ iff the Turing machine encoded by $\vec{w}$ accepts the empty input, we obtain the required undecidability result.

Let $M = (\Gamma, Q, q_0, q_1, \delta)$ be a deterministic Turing machine, where Γ is an alphabet (containing the blank symbol $\llcorner\!\lrcorner$), Q a set of states, $q_0 \in Q$ and $q_1 \in Q$ are an initial and an accepting state, respectively, and $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{-1, 1\}$ is a transition function. Computations of M can be thought of as sequences of configurations, with each configuration determined by the contents of all (infinitely many) cells of the tape, the state and the head position. We are going to encode a computation by domain elements arranged, roughly speaking, into a two-dimensional grid: one dimension is the tape and the other is time.

More precisely, we use a role T to point to the representation of the next cell on the tape (within the same configuration) and a role S to point to the representation of the same cell in a successive configuration. Concepts C_a, for $a \in \Gamma$, encode the contents of cells in the sense that a domain element belongs to C_a if the respective cell contains the symbol a. We use concepts H_q, for $q \in Q$, to indicate both the position of the head and the current state: a domain element belongs to H_q if the respective cell is under the head and the machine is in state q. We also use a concept $H_\emptyset$ to mark all other cells on the tape (that is, cells that are not under the head of the machine). Finally, roles P_{qa}, for $q \in Q$ and $a \in \Gamma$, are used to encode transitions; concepts D_σ and roles $T_{\emptyset\sigma}$, for $\sigma \in \{-1, +1\}$, to propagate the no-head marker backwards and forwards along the tape; and role T_0 to make sure the tape is initially blank beyond the input word.

Let φ be $\mathsf{DLXPath}_{\text{core}}$ node formula

$$\langle S^*\, T^*(\alpha_1 \cup \alpha_2 \cup \alpha_3 \cup \alpha_4 \cup \alpha_5 \cup \alpha_6)\rangle\,,$$

where

$$\begin{aligned}
\alpha_1 &= [\langle S^-\, T\, S \cap \bar{T}\rangle],\\
\alpha_2 &= \bigcup_{\delta(q,a)=(q',a',\sigma)} [H_q \wedge C_a \wedge \big\langle S[T^\sigma \cap \bar{P}_{q'a'}]\big\rangle],\\
\alpha_3 &= \bigcup_{a\in\Gamma} [H_\emptyset \wedge C_a \wedge \langle S[\neg C_a]\rangle],\\
\alpha_4 &= \bigcup_{\sigma\in\{-1,+1\}} [D_\sigma \wedge \langle T^\sigma \cap \bar{T}_{\emptyset\sigma}\rangle],\\
\alpha_5 &= [\langle T_0 \cap \bar{T}\rangle] \cup [\langle T\rangle \wedge \langle \bar{S}\rangle] \cup [\langle T_0^-\rangle \wedge \langle \bar{T}_0\rangle] \cup [\langle T_0^-\rangle \wedge \neg C_{\llcorner\!\lrcorner}] \cup [H_{q_1}],\\
\alpha_6 &= \bigcup_{q\in Q, a\in\Gamma} ([\langle P_{qa}^-\rangle \wedge \neg H_q] \cup [\langle P_{qa}\rangle \wedge \neg C_a]),\\
\alpha_7 &= \bigcup_{q\in Q, \sigma\in\{-1,+1\}} ([H_q \wedge \neg D_\sigma] \cup [\langle T_{\emptyset\sigma}^-\rangle \wedge \neg D_\sigma] \cup [\langle T_{\emptyset\sigma}^-\rangle \wedge \neg H_\emptyset]),
\end{aligned}$$

and $T^\sigma(y, z)$ stands for $T(y, z)$ if $\sigma = +1$ and for $T(z, y)$ if $\sigma = -1$. For every input $\vec{w} = a_1 \dots a_n \in \Gamma^*$, we take the following ABox $\mathcal{A}_{\vec{w}}$:

$$H_{q_0}(c_1), \qquad C_{a_i}(c_i) \text{ and } T(c_i, c_{i+1}),\ \text{for } 1 \leq i \leq n, \qquad T_0(c_n, c_{n+1}).$$

It is now a matter of technicality to show that $(c_1, c_1) \in \mathcal{C}ertain([\varphi], (\emptyset, \mathcal{A}_{\vec{w}}))$ iff M accepts $\vec{w}$. ❑

Theorem 5 *There exists a DL-Lite*$_{\text{core}}$ *TBox* $\mathcal{T}$ *and a* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{core}}$ *query* α *such that the problem* CERTAIN ANSWERS $(\alpha, \mathcal{T})$ *is* CONP*-hard. The problem is in* CONP *for any DL-Lite*$_{\mathcal{R}}$ *TBox* $\mathcal{T}$ *and* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ *query* α.

Proof. Let us start with the coNP-hardness of the problem. The proof will use the empty TBox. Fix the following query (note that it does not use the transitive closure operator $^+$):

$$
\begin{array}{llllllll}
\alpha = [P\big(& & & & & & & \\
(& \langle T_1[\neg A]\rangle & \wedge & \langle T_2[\neg A]\rangle & \wedge & \langle T_3[\neg A]\rangle &) & \vee \\
(& \langle T_1[\neg A]\rangle & \wedge & \langle T_2[\neg A]\rangle & \wedge & \langle F_3[A]\rangle &) & \vee \\
(& \langle T_1[\neg A]\rangle & \wedge & \langle F_2[A]\rangle & \wedge & \langle T_3[\neg A]\rangle &) & \vee \\
(& \langle T_1[\neg A]\rangle & \wedge & \langle F_2[A]\rangle & \wedge & \langle F_3[A]\rangle &) & \vee \\
(& \langle F_1[A]\rangle & \wedge & \langle T_2[\neg A]\rangle & \wedge & \langle T_3[\neg A]\rangle &) & \vee \\
(& \langle F_1[A]\rangle & \wedge & \langle T_2[\neg A]\rangle & \wedge & \langle F_3[A]\rangle &) & \vee \\
(& \langle F_1[A]\rangle & \wedge & \langle F_2[A]\rangle & \wedge & \langle T_3[\neg A]\rangle &) & \vee \\
(& \langle F_1[A]\rangle & \wedge & \langle F_2[A]\rangle & \wedge & \langle F_3[A]\rangle &) & \big)],
\end{array}
$$

where $P, T_1, T_2, T_3, F_1, F_2, F_3$ are role names and A is a concept name.

Consider the complement of the NP-complete problem 3CNF-SAT which input is a conjunction Φ of clauses of the form $\ell_1 \vee \ell_2 \vee \ell_3$, where the ℓ_i are literals, that is, propositional variables or their negations, and which output is 'yes' if Φ is not satisfiable and 'no' otherwise.

For each variable x in Φ the ABox $\mathcal{A}$ uses an individual c_x. Also, for each clause γ in Φ it uses an individual c_γ. Finally, it uses an individual c.

For each clause γ in Φ the ABox $\mathcal{A}$ contains the assertion $P(c, c_\gamma)$. Also, for each literal ℓ_i in each clause $\gamma = \ell_1 \vee \ell_2 \vee \ell_3$ of Φ the ABox $\mathcal{A}$ contains the assertion $T_i(c_\gamma, c_x)$ if $\ell_i = x$ or the assertion $F_i(c_\gamma, c_x)$ if $\ell_i = \bar{x}$.

It is a matter of technicality to show that the pair (c, c) is a certain answer to α over the KB $(\mathcal{A}, \emptyset)$ if and only if Φ is not satisfiable.

Next we carefully analyse the algorithm in the proof of Theorem 8 (which is based on the proof of Theorem 6) to obtain an coNPdata complexity algorithm which decides the problem.

Looking at the proof of Theorem 6 note that there are two sources of exponentiality in the construction of the Büchi tree automaton $\mathcal{B}_{\varphi_{\mathcal{K},\alpha}}$, for $\varphi_{\mathcal{K},\alpha}$ as in the proof of Theorem 8. First, the number of Hintikka sets $\mathcal{H}_{\varphi_{\mathcal{K},\alpha}}$ can be exponential, because it is roughly the number of all subsets of the Fisher-Ladner closure of the formula. Second, each state of the automaton has a component from $2^{\mathcal{P}_\Box(\varphi_{\mathcal{K},\alpha})}$, where $\mathcal{P}_\Box(\varphi_{\mathcal{K},\alpha}) = \{\{[\alpha]\psi, [\beta]\eta\} \mid [\alpha]\psi, [\beta]\eta \in cl(\varphi_{\mathcal{K},\alpha})\}$.

To deal with the first source, we need the following claim. Let ℓ be the fixed number of role inclusions in the TBox $\mathcal{T}$.

Claim 11.1. If the formula $\varphi_{\mathcal{K},\alpha}$ has a Hintikka tree then it has a Hintikka tree satisfying the following conditions:

1. if for an individual c_i and node formula ψ there exists a node x such that $C_i \in T(x)^{(1)}$ and $\psi \in T(x)^{(1)}$ then for any node y such that $C_i \in T(y)^{(1)}$ it holds that $\psi \in T(y)^{(1)}$;
2. for any node x such that $C_i \notin T(x)^{(1)}$ for any individual c_i, there exist at most ℓ roles R such that $\langle R\rangle C_i \in T(x)^{(1)}$.

This claim follows immediately from the construction in the proof of Proposition 11 in (Lutz and Walther 2005). Having this claim at hand we conclude that when constructing the Büchi tree automaton $\mathcal{B}_{\varphi_{\mathcal{K},\alpha}}$ we may consider not all Hintikka sets, but only those that satisfy the conditions of the claim. Since the TBox and the query are fixed, we can guess which formulas hold and construct the automaton based on the polynomial number of Hintikka sets.

To deal with the second source, note that by the construction all the states on a run of the automaton have the same $\mathcal{P}$. Hence, again, it is enough to guess this $\mathcal{P}$ and build only the corresponding part of the automaton. ❑

Theorem 6 *Checking satisfiability of a* $\mathsf{CPDL}^{(\neg)}$ *node formula can be done in* EXPTIME.

Proof. This prove goes the same lines as the proof of existence of EXPTIME algorithm for checking satisfiability of $\mathsf{PDL}^{(\neg)}$ node formula from (Lutz and Walther 2005).

In this proof we will use the shortcut $\langle\alpha\rangle\varphi$ for $\langle\alpha \cdot [\varphi]\rangle$ and $\lceil\alpha\rceil\varphi$ for $\neg\langle\alpha\rangle\neg\varphi$.[2] In fact, we even replace $\langle\alpha\rangle$ with $\langle\alpha\rangle\varphi$ and $\lceil\alpha\rceil\varphi$ in the syntax of PDL node formulas, which clearly does not change the expressive power and conciseness of formulas. Hence, we can assume that all our node formulas are in the *negation normal form*, that is the unary negation appears only in front of concept names.

Let $\Pi_0 = \{R, \neg R \mid R \text{ is a role name or inverse}\}$. It will be convenient to look at any $\mathsf{CPDL}^{(\neg)}$ path formula α not as regular expression over the alphabet $\Pi_0 \cup \{[\varphi] \mid \varphi \text{ is a node formula}\}$ but as an equivalent nondeterministic finite state automata over this alphabet with the set of states Q_α, initial state q_α, transition function δ_α and set of final states F_α. Given a state q in Q_α denote α_q the automata obtained from α by declaring q the initial state.

Fisher-Ladner closure $cl(\varphi)$ of a $\mathsf{CPDL}^{(\neg)}$ node formula φ is the smallest set closed under the following conditions:

[2]Note that the standard PDL syntax (e.g., in (Harel, Kozen, and Tiuryn 2000)) uses $\varphi?$ instead of $[\varphi]$, and $[\alpha]\varphi$ instead of $\lceil\alpha\rceil\varphi$. Since this work is influenced by the XPath query language we opted to stay faithful to its syntax and use it throughout the paper.

(C1) $\varphi \in cl(\varphi)$,

(C2) if $\psi_1 \wedge \psi_2 \in cl(\varphi)$ or $\psi_1 \vee \psi_2 \in cl(\varphi)$ then $\psi_1, \psi_2 \in cl(\varphi)$,

(C3) if $\psi \in cl(\varphi)$ then $\neg\psi \in cl(\varphi)$ (from here onwards we assume that $\neg\neg\psi$ denotes ψ, as well as $\neg\neg S$ denotes S for $S \in \Pi_0$),

(C4) if $\langle\alpha\rangle\psi \in cl(\varphi)$ then $\psi \in cl(\varphi)$, $\psi' \in cl(\varphi)$ for all $[\psi']$ in the active alphabet of α, and $\langle\alpha_q\rangle\psi \in cl(\varphi)$ for all q in Q_α,

(C5) if $\lceil\alpha\rceil\psi \in cl(\varphi)$ then $\psi \in cl(\varphi)$, $\psi' \in cl(\varphi)$ for all $[\psi']$ in the active alphabet of α, and $\lceil\alpha_q\rceil\psi \in cl(\varphi)$ for all q in Q_α.

It is straightforward to check that the size of the Fisher-Lander closure of a node formula φ is linear in the length of φ.

Given a $\mathsf{CPDL}^{(\neg)}$node formula φ, a subset Ψ of $cl(\varphi)$ is a *Hintikka set* for φ if the following conditions hold:

(H1) if $\psi_1 \wedge \psi_2 \in \Psi$ then $\psi_1 \in \Psi$ and $\psi_2 \in \Psi$,

(H2) if $\psi_1 \vee \psi_2 \in \Psi$ then $\psi_1 \in \Psi$ or $\psi_2 \in \Psi$,

(H3) $\psi \in \Psi$ if and only if $\neg\psi \notin \Psi$,

(H4) if $\lceil\alpha\rceil\psi \in \Psi$ and $q_\alpha \in F_\alpha$ then $\psi \in \Psi$,

(H5) if $\lceil\alpha\rceil\psi \in \Psi$ and $q \in \delta_\alpha(q_\alpha, [\eta])$ then $\neg\eta \in \Psi$ or $\lceil\alpha_q\rceil\psi \in \Psi$.

Denote $\mathcal{H}_\varphi$ the set of all Hintikka sets for φ.

Given a $\mathsf{CPDL}^{(\neg)}$node formula φ, let $\epsilon_1, \ldots, \epsilon_k$ be all the formulas of the form $\langle\alpha\rangle\psi$ in $cl(\varphi)$, and let Λ_φ be the set of triples $\mathcal{H}_\varphi \times (\Pi_0 \cup \{\bot\}) \times \{0, \ldots, k\}$. For every triple λ from Λ_φ denote $\lambda^{(i)}$, $1 \leq i \leq 3$, the i-th component of λ. A $(k+1)$-tuple $(\lambda, \lambda_1, \ldots, \lambda_k)$ of triples from Λ_φ is a *matching* if and only if for all $1 \leq i \leq k$ the following holds:

(M1) if $\epsilon_i = \langle\alpha\rangle\psi \in \lambda^{(1)}$ then there are $\psi_1, \ldots, \psi_n \in \lambda^{(1)}$, $n \geq 0$, such that

  (a) either $\delta_\alpha(q_\alpha, [\psi_1]\cdots[\psi_n]) \cap F_\alpha \neq \emptyset$, $\psi \in \lambda^{(1)}$, $\lambda_i^{(2)} = \bot$, and $\lambda_i^{(3)} = 0$,

  (b) or there exists $S \in \Pi_0$ and $q \in Q_\alpha$ such that $q \in \delta_\alpha(q_\alpha, [\psi_1]\cdots[\psi_n]S)$, $\epsilon_j = \langle\alpha_q\rangle\psi \in \lambda_i^{(2)}$, and $\lambda_i^{(3)} = j$;

(M2a) if $\lceil\alpha\rceil\psi \in \lambda^{(1)}$, $q \in Q_\alpha$ and $S \in \Pi_0$ are such that $q \in \delta_\alpha(q_\alpha, S)$ and $S = \lambda_i^{(2)}$ then $\lceil\alpha_q\rceil\psi \in \lambda_i^{(1)}$,

(M2b) if $\lceil\alpha\rceil\psi \in \lambda_i^{(1)}$, $q \in Q_\alpha$ and $S \in \Pi_0$ are such that $q \in \delta_\alpha(q_\alpha, S)$ and $S = (\lambda_i^{(2)})^-$ then $\lceil\alpha_q\rceil\psi \in \lambda^{(1)}$.

The condition (M2b) is the one which differs the construction from the construction in (Lutz and Walther 2005).

An *infinite k-ary M-tree* for a set M and number k is a mapping from $\{1, \ldots, k\}^*$ to M.

Given a $\mathsf{CPDL}^{(\neg)}$node formula φ with $\epsilon_1, \ldots, \epsilon_k$ set all the formulas of the form $\langle\alpha\rangle\psi$ in $cl(\varphi)$, a k-ary Λ_φ-tree T is a *Hintikka tree* for φ if and only if the following conditions hold for all nodes $x, y \in \{1, \ldots, k\}^*$:

(T1) $\varphi \in T(\varepsilon)$,

(T2) the tuple $(T(x), T(x1), \ldots, T(xk))$ is a matching,

(T3) there is no $\epsilon_i \in T(x)^{(1)}$ with $\gamma_1\gamma_2\cdots \in \{1, \ldots, k\}^\omega$ such that $\gamma_1 = i$ and $\gamma_{\ell+1} = T(x\gamma_1\ldots\gamma_\ell)^{(3)}$ for all $\ell \geq 1$,

(T4) if $\lceil\alpha\rceil\psi, \lceil\beta\rceil\eta \in T(x)^{(1)}$, $S \in \Pi_0$, $q_1 \in Q_\alpha$, and $q_2 \in Q_\beta$ are such that $q_1 \in \delta_\alpha(q_\alpha, S)$, $q_2 \in \delta_\beta(q_\beta, \neg S)$ and $\lceil\alpha_{q_1}\rceil\psi \notin T(y)^{(1)}$ then $\lceil\beta_{q_2}\rceil\eta \in T(y)^{(1)}$.

The $\mathsf{PDL}^{(\neg)}$ version of the following claim is proved in (Lutz and Walther 2005) (Proposition 11). The proof carries verbatim to the case of $\mathsf{CPDL}^{(\neg)}$.

Claim 11.2. A $\mathsf{CPDL}^{(\neg)}$ node formula is satisfiable if and only if it has a Hintikka tree.

Given a $\mathsf{CPDL}^{(\neg)}$node formula φ with k formulas of the form $\langle\alpha\rangle\psi$ in $cl(\varphi)$, denote $\mathcal{P}_\Box(\varphi) = \{\{\lceil\alpha\rceil\psi, \lceil\beta\rceil\eta\} \mid \lceil\alpha\rceil\psi, \lceil\beta\rceil\eta \in cl(\varphi)\}$. Then the k-ary Büchi tree automata $\mathcal{B}_\varphi$ over the alphabet Λ_φ is defined as follows:

1. the set Q of states is a subset of $\Lambda_\varphi \times 2^{\mathcal{P}_\Box(\varphi)} \times \{\oslash, \uparrow\}$, such that for each state $((\Psi, S, j), p, s)$ the following conditions hold:
   (a) if $\lceil\alpha\rceil\psi, \lceil\beta\rceil\eta \in \Psi$ then $\{\lceil\alpha\rceil\psi, \lceil\beta\rceil\eta\} \in p$,
   (b) if $\{\lceil\alpha\rceil\psi, \lceil\beta\rceil\eta\} \in p$, $S \in \Pi_0$, $q_1 \in \delta_\alpha(q_\alpha, S)$, $q_2 \in \delta_\beta(q_\beta, \neg S)$, and $\lceil\alpha_{q_1}\rceil\psi \notin \Psi$ then $\lceil\beta_{q_2}\rceil\eta \in \Psi$;
2. the set I of initial states is $\{((\Psi, S, j), p, s) \mid \varphi \in \Psi \text{ and } s = \oslash\}$;
3. the transition function δ is defined as follows: $((\lambda, p, s), (\Psi, S, j), (\lambda_1, p_1, s_1), \ldots, (\lambda_k, p_k, s_k)) \in \delta$ if and only if for all $1 \leq i \leq k$
   (a) $\lambda = (\Psi, S, j)$,
   (b) $p_i = p$,
   (c) $(\lambda, \lambda_1, \ldots, \lambda_k)$ is a matching, and

(d) if either $s = \oslash$, $\lambda_i^{(3)} \neq 0$ and $\epsilon_i \in \Psi$, or $s = \uparrow$, $\lambda^{(3)} = i$ and $\lambda_i^{(3)} \neq 0$, then $s_i = \uparrow$; otherwise, $s_i = \oslash$;

4. the set F of accepting states is $\{(\lambda, p, s) \mid s = \oslash\}$.

It is straightforward to check that the size of the automata $\mathcal{B}_\varphi$ is exponential in the length of φ.

Similarly to Claim , the $\mathsf{PDL}^{(\neg)}$ version of the following claim is proved in (Lutz and Walther 2005) (Proposition 15). Again, the proof carries verbatim to the case of $\mathsf{CPDL}^{(\neg)}$.

Claim 11.3. A tree T is a Hintikka tree for a $\mathsf{CPDL}^{(\neg)}$ node formula φ if and only if T is in the language of $\mathcal{B}_\varphi$.

Now the statement of the theorem follows from the fact that emptiness of a tree Büchi automaton can be checked in polynomial (quadratic) time (Vardi and Wolper 1986) and the observation that the automaton obtained by this construction is exponential in the size of the formula. ❑

Theorem 8 *The problem* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{reg}}$ CERTAIN ANSWERS OVER *DL-Lite*$_\mathcal{R}$ *is* EXPTIME*-complete. It remains* EXPTIME*-hard for* $\mathsf{DLXPath}^{\text{pos}}_{\text{reg}}$ *queries with DL-Lite*$_{\text{core}}$ *KBs, and for* $\mathsf{DLXPath}^{\text{path-pos}}_{\text{core}}$ *even with empty KBs.*

Proof. The EXPTIME-hardness immediately follows from the results of (Harel, Kozen, and Tiuryn 2000) and (Bienvenu et al. 2014). That is why next we concentrate on an algorithm.

The proof is by polynomial reduction to the satisfiability of $\mathsf{CPDL}^{(\neg)}$ formulas, which is in EXPTIME by Theorem 6. It adopts the ideas from (De Giacomo and Lenzerini 1994) to our needs. In the proof we use $\varphi \Rightarrow \psi$ as a shortcut for $\neg\varphi \vee \psi$, as well as the shortcuts $\langle\alpha\rangle\varphi$ and $\lceil\alpha\rceil\varphi$ as in the proof of Theorem 6.

Let $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ be a knowledge base, α a $\mathsf{DLXPath}^{\text{path-pos}}_{\text{core}}$ query and (c', c'') a pair of individuals. Next we construct a $\mathsf{CPDL}^{(\neg)}$ formula $\varphi_{\mathcal{K},\alpha}$ which is satisfiable if and only if $(c', c'') \notin \mathit{Certain}(\alpha, (\mathcal{A}, \mathcal{T}))$.

The formula $\varphi_{\mathcal{K},\alpha}$ uses concept and role names of $\mathcal{K}$. Besides this, $\varphi_{\mathcal{K},\alpha}$ uses a special role name Create and for every individual c_i in $\mathcal{A}$ a concept name C_i. We start with the part of $\varphi_{\mathcal{K},\alpha}$ which corresponds to the ABox $\mathcal{A}$:

$$\varphi_\mathcal{A} \quad = \quad \left(\bigwedge_{\substack{i \neq j \\ c_i, c_j \text{ individuals in } \mathcal{A}}} C_i \Rightarrow \neg C_j \right) \quad \wedge \quad \left(\bigwedge_{A(c_i) \in \mathcal{A}} C_i \Rightarrow A \right) \quad \wedge \left(\bigwedge_{R(c_i, c_j) \in \mathcal{A}} C_i \Rightarrow \langle R \rangle C_j \right).$$

Note, that if we drop UNA, then the first part of $\varphi_\mathcal{A}$ should be dropped as well. We continue with the part of $\varphi_{\mathcal{K},\alpha}$ corresponding to the TBox $\mathcal{T}$:

$$\begin{aligned} \varphi_\mathcal{T} \quad = \quad & \left(\bigwedge_{B_i \sqsubseteq B_j \,\in \mathcal{T}} \delta(B_i) \Rightarrow \delta(B_j) \right) \quad \wedge \quad \left(\bigwedge_{B_i \sqcap B_j \sqsubseteq \bot \,\in \mathcal{T}} \delta(B_i) \Rightarrow \neg\delta(B_j) \right) \\ \wedge \quad & \left(\bigwedge_{R_i \sqsubseteq R_j \,\in \mathcal{T}} \lceil \bar{R}_i \cup R_j \rceil \top \right) \quad \wedge \quad \left(\bigwedge_{R_i \sqcap R_j \sqsubseteq \bot \,\in \mathcal{T}} \lceil \bar{R}_i \cup \bar{R}_j \rceil \top \right), \end{aligned}$$

where $\delta(B)$ is $\langle R \rangle$ if $B = \exists R$ and B otherwise.

Having $\varphi_\mathcal{A}$ and $\varphi_\mathcal{T}$ components at hand, we next define the formula corresponding to the whole KB $\mathcal{K}$ with respect to the query α:

$$\begin{aligned} \varphi_\mathcal{K} \quad = \quad & \left(\bigwedge_{c_i \text{ individual in } \mathcal{A}} \langle \mathit{Create} \rangle C_i \right) \quad \wedge \quad \lceil \beta \rceil (\varphi_\mathcal{A} \wedge \varphi_\mathcal{T}) \\ \wedge \quad & \left(\bigwedge_{\substack{\psi \in cl(\varphi_\mathcal{A} \wedge \varphi_\mathcal{T} \wedge \langle \alpha \rangle \top) \\ c_i \text{ individual in } \mathcal{A}}} \langle \beta \rangle (C_i \wedge \psi) \Rightarrow \lceil \beta \rceil (C_i \Rightarrow \psi) \right), \end{aligned}$$

where

$$\beta = \left(\mathit{Create} \cup \mathit{Create}^- \cup \left(\bigcup_{R \text{ role name in } \mathcal{K}} (R \cup R^-) \right) \right)^*.$$

Finally, to take into account the testing individuals c' and c'',

$$\varphi_{\mathcal{K},\alpha} \quad = \quad \varphi_\mathcal{K} \wedge \neg \langle \mathit{Create}[C'] \alpha [C''] \rangle \top.$$

Similarly as in (De Giacomo and Lenzerini 1994) one can check that the formula $\varphi_{\mathcal{K},\alpha}$ has the required property. ❑

Theorem 9 *The problem* $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ CERTAIN ANSWERS OVER *DL-Lite*$_{\mathcal{R}}$ *is* NP-*complete. It remains* NP-*hard for DL-Lite*$_{\text{core}}$.

Proof. We start with the upper bound. In what follows we will use the following standard notions.

The *(non-oblivious) chase* $Chase(\mathcal{K})$ of a (satisfiable) KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is a set of assertions which is a result of exhaustive (possibly infinite) application the following rules, starting with $\mathcal{A}$ (by saying that a set of atoms contains $P^-(c_1, c_2)$ we mean that it contains $P(c_2, c_1)$, and saying that it contains $\exists R(c)$ we mean that there exists c' such that $R(c, c')$ is contained in this set):

– add $A_i(c)$ if $\mathcal{T}$ contains $B \sqsubseteq A_i$ and $Chase(\mathcal{K})$ contains $B(c)$;

– add $R_2(c_1, c_2)$ if $\mathcal{T}$ contains $R_1 \sqsubseteq R_2$ and $Chase(\mathcal{K})$ contains $R_1(c_1, c_2)$;

– add $R(c, c^{new})$, for a fresh c^{new}, if $\mathcal{T}$ contains $B \sqsubseteq \exists R$ and $Chase(\mathcal{K})$ contains $B(c)$ but does not contain $\exists R(c)$.

The *canonical model* $Can(\mathcal{K})$ of a satisfiable KB $\mathcal{K}$ is the model which interprets all individuals in the $Chase(\mathcal{K})$ by themselves, $c \in A^{Can(\mathcal{K})}$ if and only if $A(c) \in Chase(\mathcal{K})$, and $(c_1, c_2) \in P^{Can(\mathcal{K})}$ if and only if $P(c_1, c_2) \in Chase(\mathcal{K})$. It is well known that for any model $\mathcal{I}$ of $\mathcal{K}$ there exists a homomorphism from $Can(\mathcal{K})$ to $\mathcal{I}$.

The following claim can be proved by straightforward induction on the structure of the $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ query.

Claim 11.4. A pair of individuals (c_1, c_2) is a certain answer to a $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ query α over a satisfiable KB $\mathcal{K}$ if and only if $(c_1, c_2) \in [\![\alpha]\!]^{Can(\mathcal{K})}$.

Essentially, this claim says that we can concentrate only on the canonical model. Since it can be quite big and even infinite, next we show that for positive query answering it is always enough to guess just a polynomial part of the canonical model. In what follows we need one more definition.

Given a $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ query which does not use unary $\vee$ primitive, binary $\cup$ primitive, concept names, as well as the transitive closure operator $+$ but may use the reflexive transitive closure operator $*$ over roles, a *skeleton* $\mathcal{S}_\alpha$ of α is a directed unordered tree represented the structure of α in the sense that besides the root it has a distinguished node called *drain*, each edge is labelled by either R or R^* according to α and branching in this tree represents nesting $[\varphi]$ in α.

Let now α be a $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ query and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a *DL-Lite*$_{\mathcal{R}}$ KB. Without loss of generality we may assume that

– the KB $\mathcal{K}$ is satisfiable (if not, it can be easily checked in polynomial time),

– the vocabulary does not contain any concept names (they can be simulated by role names in the straightforward manner),

– α does not use $\vee$ and $\cup$ primitives (our algorithm can always guess which alternative to use), and

– α does not use the transitive closure operator $+$, but may use the reflexive transitive closure operator $*$ over roles (these primitives are interreducible).

For clarity of exposition we also assume that $\mathcal{K}$ uses only one individual c, the generalisation to arbitrary case is a matter of technicality.

It is straightforward to show that $(c, c) \in [\![\alpha]\!]^{Can(\mathcal{K})}$ if and only if there exists an *embedding* γ of the skeleton $\mathcal{S}_\alpha$ to $Can(\mathcal{K})$, that is a mapping preserving labels in the way that

– if R is a label of (a, a') in $\mathcal{S}_\alpha$ then $R^{Can(\mathcal{K})}(\gamma(a), \gamma(a'))$ holds,

– if R^* is a label of (a, a') in $\mathcal{S}_\alpha$ then there exist a sequence $c_1, \ldots, c_n$, $n \geq 1$, without repetitions such that $c_1 = \gamma(a)$, $c_n = \gamma(a')$ and $R^{Can(\mathcal{K})}(\gamma(c_i), \gamma(c_{i+1}))$ holds for all i,

which maps the root and the drain to c. Note that given γ, under our assumptions the sequence $c_1, \ldots, c_n$ (without repetitions) satisfying the second requirement of embedding is unique for every edge (a, a') labelled R^*. We denote it $\Gamma(a, a')$.

Having this fact at hand, it is enough to show that if such an embedding γ exists then there exists an embedding such that the maximal distance from c to the image of a node from the skeleton in $Can(\mathcal{K})$ is of polynomial size. Indeed, if it holds than we can guess all the polynomial number of branches in $Can(\mathcal{K})$ which leads to images of all the nodes from skeleton and check whether the query holds on this polynomially sized sub-interpretation of the canonical model. That is why the rest of the proof is devoted to reducing this maximal distance for a given embedding. For this, we reuse the *shortcutting* technique, developed in (Benedikt, Fan, and Geerts 2008) for a similar problem in the XML context.

Let c' be the image $\gamma(a')$ of a node in $\mathcal{S}_\alpha$ with the maximal distance from c in $Can(\mathcal{K})$. The branch $c = c_1, \ldots, c_m = c'$ in the canonical model can be partitioned in the following way: c_i and c_{i+1} are in the same part if and only if

– both of them are not images of any nodes of $\mathcal{S}_\alpha$ by γ, and

– $\{(a, a') \mid c_i \in \Gamma(a, a')\} = \{(a, a') \mid c_{i+1} \in \Gamma(a, a')\}$.

That is, every part of this partitioning consists of a continuos sequence of nodes on the branch from c to c', and every image of a node of the skeleton (which is on the branch) forms its own part. However, we are interested in those parts which contains more than $|\mathcal{T}|$ elements. If such a part does not exist, we are done, because the length n of the branch is bounded by $|\mathcal{T}| \times |\mathcal{S}_\alpha|$. If, contrary, such a part exists, then it contains two elements c_i, c_j, $i > j$, with the same type in $Can(\mathcal{K})$ (by *type* of an element

d in an interpretation $\mathcal{I}$ we mean the set $\{(\exists R) \mid$ there exists d' such that $R^{\mathcal{I}}(d, d')\}$). Since the types are the same, we can simply replace the tree-like section of the canonical model starting in c_i with the section starting in c_j. After such a replacement the canonical model stays the same (since it is infinite), but the distance from c to c' strictly decreases.

Applying such a procedure to the canonical model $Can(\mathcal{K})$ and embedding γ while possible we arrive to an embedding with desired properties. It means that the pair (c, c) is a certain answer to α over satisfiable $\mathcal{K}$ if and only if there exists a polynomially sized sub-interpretation of the canonical model of $\mathcal{K}$ which witness α for (c, c).

To show hardness of $\mathsf{DLXPath}^{\mathrm{pos}}_{\mathrm{core}}$ evaluation over *DL-Lite*$_{\mathrm{core}}$ we give a reduction from 3CNF-SAT. Suppose we are given a conjunction Φ of clauses of the form $\ell_1 \vee \ell_2 \vee \ell_3$, where the ℓ_k are literals, that is, propositional variables or their negations (we can assume that all literals in each clause are distinct). Let $x_1, \dots, x_n$ be the variables in Φ.

Consider the KB $\mathcal{K}$ with the ABox $\mathcal{A} = \{A(c)\}$ and the TBox $\mathcal{T}$ consisting of the inclusions

$$\begin{array}{rcll} A & \sqsubseteq & \exists(X_1^v)^-, & \text{for } v \in \{\top, \bot\}, \\ \exists X_{i-1}^v & \sqsubseteq & \exists(X_i^u)^-, & \text{for } v, u \in \{\top, \bot\} \text{ and } 1 < i \leq n. \end{array} \tag{2}$$

For every variable x_i and truth value $v \in \{\top, \bot\}$ let $\psi^v_{x_i}$ be the node formula:

$$\langle (X_n^\top \cup X_n^\bot) \cdots (X_{i+1}^\top \cup X_{i+1}^\bot) \cdot X_i^v \cdot (X_{i-1}^\top \cup X_{i-1}^\bot) \cdots (X_1^\top \cup X_1^\bot) \rangle.$$

Let φ be a node formula $\langle \alpha[\psi] \rangle$, where

$$\alpha = ((X_1^\top)^- \cup (X_1^\bot)^-) \cdot \ldots \cdot ((X_n^\top)^- \cup (X_n^\bot)^-),$$

and ψ is a conjunction of node formulas of the following form, for each clause γ in Φ, using variables x, y, z:

$$\bigvee_{\substack{(v_x, v_y, v_z) \text{ assignment of } (x,y,z) \\ \text{satisfying } \gamma}} \psi_x^{v_x} \wedge \psi_y^{v_y} \wedge \psi_z^{v_z}.$$

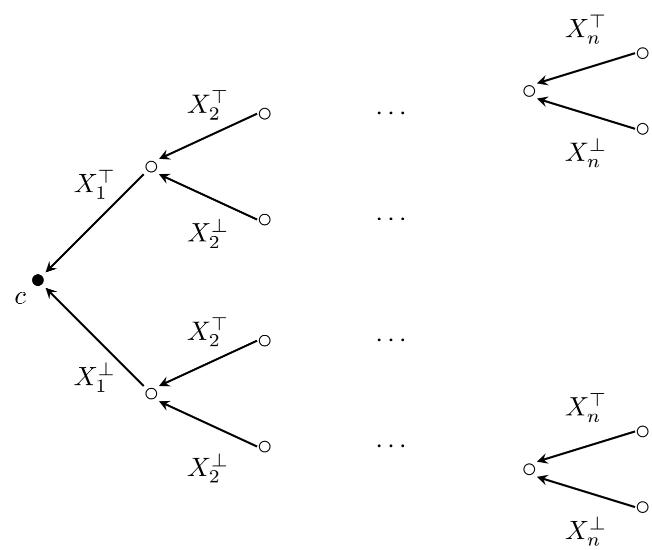
We now show that the pair (c, c) is a certain answer to $[\varphi]$ over $\mathcal{K}$ if and only if Φ is satisfiable.

Suppose first that Φ is satisfiable and let $\sigma : \{x_1, \dots, x_n\} \to \{\top, \bot\}$ be a truth assignment such that $\sigma(\Phi)$ is true. Take any model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}$. Then using assertions from (2) we can find elements $d_0, d_1, d_2, \dots, d_n \in \Delta^{\mathcal{I}}$ such that:

$$\begin{array}{l} d_0 = c^{\mathcal{I}}, \text{ and} \\ (X_i^{\sigma(x_i)})^{\mathcal{I}}(d_i, d_{i-1}), \text{ for all } 1 \leq i \leq n. \end{array} \tag{3}$$

Then by definition $(c^{\mathcal{I}}, d_n) \in [\![\alpha]\!]^{\mathcal{I}}$. To show that $c^{\mathcal{I}} \in [\![\varphi]\!]^{\mathcal{I}}$ it suffices to check that $d_n \in [\![\psi]\!]^{\mathcal{I}}$. To see this take any clause γ in Φ using variables x, y, z and consider the formula $\psi_x^{\sigma(x)} \wedge \psi_y^{\sigma(y)} \wedge \psi_z^{\sigma(z)}$. The facts (3) above immediately imply that $d_n \in [\![\psi_x^{\sigma(x)}]\!]^{\mathcal{I}}$. Analogously we get that $d_n \in [\![\psi_y^{\sigma(y)}]\!]^{\mathcal{I}}$ and $d_n \in [\![\psi_z^{\sigma(z)}]\!]^{\mathcal{I}}$. From this it follows that $d_n \in [\![\psi]\!]^{\mathcal{I}}$ and therefore $c^{\mathcal{I}} \in [\![\varphi]\!]^{\mathcal{I}}$. Since $\mathcal{I}$ was chosen arbitrary we conclude that (c, c) is a certain answer to $[\varphi]$ over $\mathcal{K}$.

Conversely, assume that (c, c) is a certain answer to $[\varphi]$ over $\mathcal{K}$. Consider the canonical model $\mathcal{I}_0$ of $\mathcal{K}$. Therefore we can view $\mathcal{I}_0$ as a complete (inverted) binary tree of height n, as illustrated in the following image.



As before we depict the fact that $(X_i^v)^{\mathcal{I}_0}(d,d')$ holds by having an X_i^v labelled arrow between nodes representing d and d'. Since (c,c) is a certain answer to $[\varphi]$ over $\mathcal{K}$ we have that $c^{\mathcal{I}_0} \in [\![\varphi]\!]^{\mathcal{I}_0}$. Therefore there exist elements $d_0, d_1, \ldots, d_n$ in $\mathcal{I}_0$ and truth values $v_1, \ldots, v_n$ such that:

$$\begin{array}{l} d_0 = c^{\mathcal{I}_0}, \\ (X_i^{v_i})^{\mathcal{I}_0}(d_i, d_{i-1}), \text{ for all } 1 \leq i \leq n, \text{ and} \\ d_n \in [\![\psi]\!]^{\mathcal{I}_0}. \end{array}$$

Define now $\sigma(x_i) := v_i$. for all $i = 1, \ldots, n$. We claim that σ is a satisfying assignment for Φ. To see this take any clause γ in Φ and assume that γ uses variables x, y, z. Since $d_n \in [\![\psi]\!]^{\mathcal{I}_0}$ this implies that there is an assignment v_x, v_y, v_z of variables x, y, z satisfying γ and such that the formula $\psi_x^{v_x} \wedge \psi_y^{v_y} \wedge \psi_y^{v_y}$ is true at d_n. In particular $\psi_x^{v_x}$ is true at d_n which implies that we can reach $c^{\mathcal{I}_0}$ using an $X_i^{v_x}$ labelled edge, where $x = x_i$. Since every path in $\mathcal{I}_0$ has a unique label we conclude that $v_x = v_i$ and similarly for y and z. Since γ was arbitrary we conclude that Φ is satisfiable. ❑

Theorem 10 *Answering* $\mathsf{DLXPath}_{\text{reg}}^{\text{path-pos}}$ *queries over* $\mathcal{ELHI}_\bot$ *KBs is* EXPTIME-*complete in combined complexity, with the lower bound already holding for* $\mathcal{EL}$ *and* $\mathsf{DLXPath}_{\text{reg}}^{\text{pos}}$. *In data complexity the problem is* CONP-*complete for* $\mathsf{DLXPath}_{\text{reg}}^{\text{path-pos}}$ *and drops to* PTIME-*complete for* $\mathsf{DLXPath}_{\text{reg}}^{\text{pos}}$, *again for both* $\mathcal{ELHI}_\bot$ *and* $\mathcal{EL}$.

Proof.

We begin with the EXPTIME bounds. These require a slight modification to the proof of Lemma 7, to make it work for $\mathcal{ELHI}_\bot$ knowledge bases. Indeed, the EXPTIME-hardness remains, as it is shown even for emtpy KB's. To show the EXPTIME upper bound, all we need to show is how to modify the CPDL$^{(\neg)}$ formula $\phi_{\mathcal{K},a}$ so that it correctly verifies the assertions in our TBox $\mathcal{T}$. According to the normal form for $\mathcal{ELHI}_\bot$ knowledge bases, we need to include into $\phi_{\mathcal{K},a}$ conjuncts for assertions of form

$$A \sqsubseteq \bot \qquad A \sqsubseteq \exists R.B \qquad \top \sqsubseteq A$$

$$B_1 \sqcap B_2 \sqsubseteq A \qquad \exists R.B \sqsubseteq A.$$

We thus redefine the part of $\phi_{\mathcal{K},a}$ that corresponds to $\mathcal{T}$, as follows:

$$\begin{aligned} \varphi_{\mathcal{T}} \quad = \quad & \left(\bigwedge_{A \sqsubseteq \bot \,\in\, \mathcal{T}} \neg A \right) \wedge \left(\bigwedge_{\top \sqsubseteq A \,\in\, \mathcal{T}} A \right) \wedge \left(\bigwedge_{B_i \sqcap B_j \sqsubseteq A \,\in\, \mathcal{T}} (\delta(B_i) \wedge \delta(B_j)) \Rightarrow A \right) \\ & \wedge \left(\bigwedge_{A \sqsubseteq \exists R.B \,\in\, \mathcal{T}} A \Rightarrow \langle R \rangle \delta(B) \right) \wedge \left(\bigwedge_{\exists R.B \sqsubseteq A \,\in\, \mathcal{T}} \langle R \rangle \delta(B) \Rightarrow A \right), \end{aligned}$$

where $\delta(B)$ is $\langle R \rangle$ if $B = \exists R$ and B otherwise.

The proof now follows just as in Lemma 7.

Furthermore, CONP-hardness follows from the proof of Theorem 5, as the reduction given there uses an empty TBox. Membership in CONP follows again by following the proof of Theorem 5, taking, of course, the modification given above into consideration.

PTIME-completeness follows immediately from PTIME-completeness of nested two way regular path queries given in (Bienvenu et al. 2014). ❑

Theorem 11 *Answering* $\mathsf{DLXPath}_{\text{core}}^{\text{pos}}$ *queries over* $\mathcal{ELH}_\bot$ *KBs is* NP-*complete in combined complexity. For* $\mathcal{ELHI}_\bot$ *KBs it becomes* EXPTIME-*complete. For both DLs the problem is* PTIME-*complete in data complexity. Apart from combined complexity for* $\mathcal{ELHI}_\bot$ *KBs the lower bounds already hold for* $\mathcal{EL}$.

Proof. We begin by proving the results for $\mathcal{ELH}_\bot$. To show that the certain answers problem is NP-hard for $\mathsf{DLXPath}_{\text{core}}^{\text{pos}}$ even over $\mathcal{EL}$ we use a construction similar to the one in the proof of Theorem 9. We again give a reduction from the 3CNF-SAT problem, this time using the KB $\mathcal{K}$ with the ABox $\mathcal{A} = \{A(c)\}$ and the TBox $\mathcal{T}$ consisting of the inclusions

$$\begin{array}{rcll} A & \sqsubseteq & \exists X_1^v.B_1, & \text{for } v \in \{\top, \bot\}, \\ B_{i-1} & \sqsubseteq & \exists X_i^v.B_i, & \text{for } v \in \{\top, \bot\} \text{ and } 1 < i \leq n. \end{array} \tag{4}$$

For x_i and truth value $v \in \{\top, \bot\}$ the formula $\psi_{x_i}^v$ is now defined as:

$$\langle ((X_n^\top)^- \cup (X_n^\bot)^-) \cdots ((X_{i+1}^\top)^- \cup (X_{i+1}^\bot)^-) \cdot (X_i^v)^- \cdot ((X_{i-1}^\top)^- \cup (X_{i-1}^\bot)^-) \cdots ((X_1^\top)^- \cup (X_1^\bot)^-) \rangle.$$

We define φ to be a formula $\langle \alpha[\psi] \rangle$, where

$$\alpha = (X_1^\top \cup X_1^\perp) \cdot \ldots \cdot (X_n^\top \cup X_n^\perp),$$

and ψ is the same as above.

The idea here is that the canonical model is now an ordinary binary tree with the root labelled A and each of the nodes on level i labelled by B_i. The edges going from level i to the level $i+1$ of the tree are labelled X_i^v, and not $(X_i^v)^-$ as before. Checking that the reduction works as intended can be done analogously as in the case of *DL-Lite*.

For the NP upper bound we show how the algorithm for the case of *DL-Lite*$_\mathcal{R}$ can be extended to $\mathcal{ELH}_\perp$. We modify the chase procedure $Chase(\mathcal{K})$ from the proof of Theorem 9 by stating that it starts with assertions in $\mathcal{A}$ (and saying that it contains $\exists R.B(c)$ when we have c' such that $R(c, c')$ and $B(c')$ is contained in this set) and uses the following rules:

- add $A(c)$ if $\mathcal{T}$ contains $\exists R.B \sqsubseteq A$ and $Chase(\mathcal{K})$ contains $\exists R.B(c)$;
- add $R(c, c^{new})$ and $B(c^{new})$, for a fresh c^{new}, if $\mathcal{T}$ contains $A \sqsubseteq \exists R.B$ and $Chase(\mathcal{K})$ contains $A(c)$ but does not contain $\exists R.B(c)$;
- add $B(c)$ if $\mathcal{T}$ contains $A_1 \sqcap A_2 \sqsubseteq B$ and $Chase(\mathcal{K})$ contains $A_1(c)$ and $A_2(c)$.

Note that here we use the normal form for $\mathcal{ELH}_\perp$ which is the same as the normal form for $\mathcal{ELHI}_\perp$, but without allowing inverses of roles. It is easy to check that the canonical model produced by these rules has the same properties as in the proof of Theorem 9. Since the remainder of the NP algorithm there only uses the fact that KB satisfiability can be checked in PTIME which holds for $\mathcal{ELH}_\perp$ (Stefanoni, Motik, and Horrocks 2013) we obtain the desired bound.

Next we show that the combined complexity of query answering for $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ becomes EXPTIME-complete if we use $\mathcal{ELHI}_\perp$ knowledge bases by reducing the problem of $\mathcal{ELHI}_\perp$ satisfiability to query answering.

Assume we are given a $\mathcal{ELHI}_\perp$ knowledge base $\mathcal{K} = (\mathcal{A}, \mathcal{T})$. Define a query α as $\alpha := [A]$.

Next, take a knowledge base $\mathcal{K}' = (\mathcal{A} \cup \{C(c_0), \mathcal{T} \cup \{A \sqsubseteq \perp\}\})$, where A and C are fresh concepts not appearing in $\mathcal{K}$ and c_0 a fresh individual.

It is straightforward to check that (c_0, c_0) is not a certain answer to α over $\mathcal{K}'$ if and only if $\mathcal{K}$ is satisfiable. Since the problem of $\mathcal{ELHI}_\perp$ satisfiability is known to be EXPTIME-complete (Baader, Brandt, and Lutz 2008) we have established the lower bound. The fact that the problem belongs to EXPTIME follows from Theorem 10.

The bound for data complexity follows from (Bienvenu et al. 2014) and the fact that every $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$ query can be simulated by a N2RPQ (Libkin, Martens, and Vrgoč 2013). ❑