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 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 knowledge bases. In particular, not much is known about the interplay between DLs and XPath. The latter 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 simple DLs like 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 shape of negation allowed in the queries we can arrive at XPath fragments that have a potential to be used in practice.

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) and their practical counterpart the Web Ontology Language (OWL) (Motik et al. 2012), such as 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 relevant to DLs database paradigm 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 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. It is worthwhile noting that connections between XML and DLs have been explored before (Calvanese, De Giacomo, and Lenzerini 1999), however this work did not consider querying knowledge bases, but it instead concentrated on reasoning about their structural properties.

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 a binary predicate representing direct flights, another one connecting 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 negative 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 which is not available in languages like 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 cause difficulties even when studied in isolation.

The first step towards introducing 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 query answers can be computed within reasonable complexity. To that extent, we first adapt the XPath query language to work over ontologies, arriving at DLXPath—an expressive language designed specifically for DL knowledge bases. To get a feeling for the interplay between DLXPath and DLs, we then study the problem of evaluating DLXPath queries over *DL-Lite*$_\mathcal{R}$ and *DL-Lite*$_{core}$ ontologies. Finally, we show how most of our techniques can then be applied to study query answering for DLs 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 pose 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 shape of allowed negation we obtain expressive languages whose queries can be answered in coNP and even NLogSpace, both over *DL-Lite*$_\mathcal{R}$ and *DL-Lite*$_{core}$. We then move to studying the *combined complexity*, where both the knowledge base and the 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. Finally, we also show how our techniques can be extended to work for 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.

Preliminaries

The language of *DL-Lite*$_\mathcal{R}$ (and *DL-Lite*$_{core}$) (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*$_{core}$ 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:

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

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 one to use *DL-Lite* as a foundation for practical data management applications, such as 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 DLXPath for querying knowledge bases to be useful in practical applications.

In what follows, we will consider several fragments of 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*

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

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 DLXPath path formula. The *certain answers* of α over $\mathcal{K}$, denoted $\mathcal{C}ertain(\alpha, \mathcal{K})$, 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 $\mathcal{C}ertain(\varphi, \mathcal{K})$ for a 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 HasDirectFlight, which connects cities with a direct flight, and HasUNESCOSite, which connects cities with their UNESCO world heritage sites, as well as concept name InDanger denoting that a particular site is endangered. Let KB $\mathcal{K}$ represent 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* belongs to $\mathcal{C}ertain(\varphi, \mathcal{K})$, 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.

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

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

As already noted, the formalism of core XPath forms the basis of most query languages over 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* 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

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

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 pose more complex queries than the ones in the core fragment. Consider again the knowledge base $\mathcal{K}$ from Example 2. We can now ask whether 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 $(Liverpool, Jerusalem)$ is in $\mathcal{C}ertain(\alpha, \mathcal{K})$, 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, provided Cultural is the corresponding concept name in $\mathcal{K}$.

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

Before continuing 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 possible 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 two sections we analyse the complexity of evaluating DLXPath queries over *DL-Lite* KBs.

Data Complexity of DLXPath Query Evaluation

As 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 $(c_1, c_2) \in \mathcal{C}ertain(\alpha, (\mathcal{T}, \mathcal{A}))$?

As was previously mentioned, the 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 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* $\mathsf{DLXPath}_{\text{core}}$ *query* α *such that the problem* CERTAIN ANSWERS $(\alpha, \emptyset)$ *is undecidable (that is, certain answers is already undecidable when using an ampty TBox).*

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 $\mathsf{DLXPath}_{\text{core}}^{\text{path-pos}}$ 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 $\mathsf{DLXPath}_{\text{reg}}^{\text{path-pos}}$ accordingly. We then have the following theorem.

Theorem 5 *There exists a* $\mathsf{DLXPath}_{\text{core}}^{\text{path-pos}}$ *query* α *such that the problem* CERTAIN ANSWERS $(\alpha, \emptyset)$ *is* CONP*-hard. The problem is in* CONP *for any DL-Lite*$_\mathcal{R}$ *TBox* $\mathcal{T}$ *and* $\mathsf{DLXPath}_{\text{reg}}^{\text{path-pos}}$ *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 $\mathsf{DLXPath}_{\text{core}}^{\text{pos}}$ and $\mathsf{DLXPath}_{\text{reg}}^{\text{pos}}$, 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*$_{core}$ and *DL-Lite*$_\mathcal{R}$. Furthermore, since they can express graph reachability, the problem is already NLOGSPACE hard even for $\mathsf{DLXPath}_{\text{core}}^{\text{pos}}$ queries (Jones 1975). 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'.

$\mathsf{DLXPath}_y^z$ CERTAIN ANSWERS OVER *DL-Lite*$_X$	
Input:	*DL-Lite*$_X$ KB $\mathcal{K}$, $\mathsf{DLXPath}_y^z$ 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 the 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 $\mathsf{DLXPath}_{\text{reg}}^{\text{path-pos}}$ 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 $\mathsf{DLXPath}_{\text{reg}}^{\text{path-pos}}$ (i.e., CPDL), except that it does not allow the inverse P_j^- of role names as path expressions. The standard problem in the PDL community is *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 $\mathit{PDL}^{\neg}$) 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 $\mathit{PDL}^{(\neg)}$, 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 $\mathsf{DLXPath}_{\text{core}}^{\text{path-pos}}$, even for empty KBs. Also, in (Bienvenu et al. 2014) hardness was established even for $\mathsf{DLXPath}_{\text{reg}}^{\text{pos}}$.

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

The only remaining fragment is that of $\mathsf{DLXPath}_{\text{core}}^{\text{pos}}$ 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* $\mathsf{DLXPath}_{\text{core}}^{\text{pos}}$ CERTAIN ANSWERS OVER *DL-Lite*$_{\mathcal{R}}$ *is* NP*-complete. It remains* NP*-hard for DL-Lite*$_{\text{core}}$.

It is worthwhile 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 in combined complexity of the problem lies somewhere very close to here, leaving 2RPQs on one side and $\mathsf{DLXPath}_{\text{core}}^{\text{pos}}$ 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*$_{core}$ 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.

DLXPath for $\mathcal{EL}$ Family

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 for 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 defined by the grammar

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

where A_i ranges over concept names and R over roles. Then, 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*$_{\mathcal{R}}$. TBoxes of $\mathcal{ELH}_\bot$ disallows role inverses, and plain $\mathcal{EL}$ further disallows role inclusions and $\bot$ in concept inclusions.

The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ is extended to $\mathcal{ELHI}_\bot$ concepts similarly to the case of *DL-Lite*, with $\top^{\mathcal{I}}$ denoting the universal relation, $(C_1 \sqcap C_2)^{\mathcal{I}}$ the intersection of $C_1^{\mathcal{I}}$ and $C_2^{\mathcal{I}}$, and

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

Similarly, the satisfaction relation $\models$ extends to $\mathcal{ELHI}_\bot$ inclusions in the standard way.

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. Since the proof of Theorem 4 uses an empty TBox this 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 any DL between* $\mathcal{ELHI}_\bot$ *and* $\mathcal{EL}$.

The last remaining fragment we study is $\mathsf{DLXPath}^{\text{pos}}_{\text{core}}$. Here we observe that comparing to *DL-Lite* 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 (Calvanese et al. 2006). The NP bounds for combined complexity showed for *DL-Lite*$_{\mathcal{R}}$ hold for $\mathcal{ELH}_\bot$ and $\mathcal{EL}$ 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 all the lower bounds already hold for* $\mathcal{EL}$.

Conclusions and Future Work

In this paper we conducted a detailed study of 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 shape of allowed 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.

Acknowledgements. We thank Diego Figueira for his comments. Reutter and Vrgoč are supported by the Millennium Nucleus Center for Semantic Web Research Grant NC120004.

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. 2006. Data complexity of query answering in description logics. In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, 260–270.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *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.; De Giacomo, G.; and Lenzerini, M. 1999. Representing and reasoning on XML documents: A description logic approach. *J. Log. Comput.* 9(3):295–318.

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/`.

Jones, N. D. 1975. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*.

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.

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.