Frontiers and Exact Learning of $\mathcal{ELI}$ Queries under DL-Lite Ontologies

Maurice Funk$^1$, Jean Christoph Jung$^2$ and Carsten Lutz$^1$

$^1$ Leipzig University, Faculty of Mathematics and Computer Science, Germany
$^2$ University of Hildesheim, Institute of Computer Science, Germany
mfunk@informatik.uni-leipzig.de, jungj@uni-hildesheim.de, clu@informatik.uni-leipzig.de

Abstract

We study $\mathcal{ELI}$ queries (ELIQs) in the presence of ontologies formulated in the description logic DL-Lite. For the dialect DL-Lite$^3$, we show that ELIQs have a frontier (set of least general generalizations) that is of polynomial size and can be computed in polynomial time. In the dialect DL-Lite$^2$, in contrast, frontiers may be infinite. We identify a natural syntactic restriction that enables the same positive results as for DL-Lite$^3$. We use our results on frontiers to show that ELIQs are learnable in polynomial time in the presence of a DL-Lite$^4$/restricted DL-Lite$^2$ ontology in Angluin’s framework of exact learning with only membership queries.

1 Introduction

In the widely studied paradigm of ontology-mediated querying, a database query is enriched with an ontology that provides domain knowledge as well as additional vocabulary for query formulation [Bienvenu et al., 2013b; Calvanese et al., 2009]. We consider ontologies formulated in description logics (DLs) of the DL-Lite family and queries that are $\mathcal{ELI}$ queries (ELIQs) or, in other words, tree-shaped unary conjunctive queries (CQs). DL-Lite is a prominent choice for the ontology language as it underpins the OWL 2 QL profile of the OWL ontology language [OWL Working Group, 2009]. Likewise, ELIQs are a prominent choice for the query language as they are computationally very well-behaved: without an ontology, they can be evaluated in polynomial time in combined complexity, in contrast to NP-completeness for unrestricted CQs. Moreover, in the form of $\mathcal{ELI}$ concepts they are a central building block of ontologies in several dialects of DL-Lite and beyond.

The aim of this paper is to study the related topics of computing least general generalizations (LGGs) of ELIQs under DL-Lite ontologies and learning ELIQs under DL-Lite ontologies in Angluin’s framework of exact learning [Angluin, 1987a; Angluin, 1987b]. Computing generalizations is a natural operation in query engineering that plays a crucial role in learning logical formulas [Plotkin, 1970; Muggleton, 1991], in particular in exact learning [ten Cate and Dalmau, 2021]. Exact learning, in turn, is concerned with constructing queries and ontologies. This can be challenging and costly, especially when logic expertise and domain knowledge are not in the same hands. Aiming at such cases, exact learning provides a systematic protocol for query engineering in which a learner interacts in a game-like fashion with an oracle, which may be a domain expert.

Our results on LGGs concern the notion of a frontier of an ELIQ $q$ w.r.t. an ontology $O$. Such a frontier is a set $F$ of ELIQs that generalize $q$, that is, $q \subseteq_\mathcal{O} q_F$ and $q_F \not\subseteq_\mathcal{O} q$ for all $q_F \in F$, where $\subseteq_\mathcal{O}$ denotes query containment w.r.t. $O$. Moreover, $F$ must be complete in that for all ELIQs $q'$ with $q \subseteq_\mathcal{O} q'$ and $q' \not\subseteq_\mathcal{O} q$, there is a $q_F \in F$ such that $q_F \subseteq_\mathcal{O} q'$. We are interested in computing a frontier that contains only polynomially many ELIQs of polynomial size, in polynomial time. This is possible in the case of ELIQs without ontologies as shown in [ten Cate and Dalmau, 2021]; for the simpler $\mathcal{EL}$ queries, the same had been observed earlier (also without ontologies) in [Baader et al., 2018; Krikel, 2019]. In contrast, unrestricted CQs do not even admit finite frontiers [Nesetril and Tardif, 2000].

In exact learning, the learner and the oracle know and agree on the ontology $O$, and they also agree on the target query $q_T$ to use only concept and role names from $O$. The learner may ask membership queries where they produce an ABox $A$ and a candidate answer $a$ and ask whether $A, O \models q_T(a)$, that is, whether $a$ is an answer to $q_T$ w.r.t. $O$ on $A$. The oracle faithfully answers “yes” or “no”. Polynomial time learnability then means that the learner has an algorithm for constructing $q_T$, up to equivalence w.r.t. $O$, with running time bounded by a polynomial in the sizes of $q_T$ and $O$.

Learning with only membership queries, as described above and studied in this article, is a strong form of exact learning. In fact, there are not many cases where polynomial time learning with only membership queries is possible, ELIQs without ontologies being an important example [ten Cate and Dalmau, 2021]. Often, one would therefore also admit equivalence queries where the learner provides a hypothesis $\mathcal{ELI}$ query $q_H$ and asks whether $q_H$ is equivalent to $q_T$ under $O$; the oracle answers “yes” or provides a counterexample, that is, an ABox $A$ and answer $a$ such that $A, O \models q_T(a)$ and $A, O \not\models q_H(a)$ or vice versa. This is done, for instance, in [Konev et al., 2018; Funk et al., 2021].
DL-Lite$^F$, equipped with role inclusions (also known as role hierarchies) and functional roles, respectively. Both dialects admit concept and role disjointness constraints and ELI concepts on the right-hand side of concept inclusions [Calvanese et al., 2007; Kikot et al., 2011]. We show that DL-Lite$^H$ admits polynomial frontiers that can be computed in polynomial time, and that DL-Lite$^F$ does not even admit finite frontiers. We then introduce a fragment DL-Lite$^F$ of DL-Lite$^F$ that restricts the use of inverse functional roles on the right-hand side of concept inclusions and show that it is as well-behaved as DL-Lite$^H$. Both frontier constructions require a rather subtle analysis. We also note that adding conjunction results in DL-Lite$^H$ side of concept inclusions and show that it is as well-behaved as DL-Lite$^H$. We then introduce a fragment DL-Lite$^H$ of DL-Lite$^F$ that is, an ELIQ logics formulated in (nomially many data examples of the form (A, a), labeled data examples [Funk et al., 2022]. The learning algorithm uses only membership queries, and that they cannot be learned with only polynomially many membership queries when conjunction is admitted.

Proof details are given in the long version [Funk et al., 2022].

Related Work. Exact learning of queries in the context of description logics has been studied in [Funk et al., 2021] while [Konev et al., 2018] considers learning entire ontologies, see also [Ozaki et al., 2020; Ozaki, 2020]. It is shown in [Funk et al., 2021] that a restricted form of CQs (that do not encompass all ELIQs) can be learned in polynomial time under EL ontologies using both membership and equivalence queries. The results from that paper indicate that inverse roles provide a challenge for exact learning under ontologies and thus it is possible that we can handle them without any restrictions in our context. Related forms of learning are the construction of the least common subsumer (LCS) and the most specific concept (MSC) [Baader, 2003; Baader et al., 1999; Baader et al., 2007; Jung et al., 2020b; Zarrieß and Turhan, 2013] which may both be viewed as a form of query generalization. There is also a more loosely related research thread on learning DL concepts from labeled data examples [Funk et al., 2019; Jung et al., 2020a; Lehmann and Hitzler, 2010; Lehmann and Vökel, 2014; Sarker and Hitzler, 2019].

2 Preliminaries

Ontologies and ABoxes. Let $N_C$, $N_R$, and $N_I$ be countably infinite sets of concept, role, and individual names. A role $R$ is a role name $r \in N_R$ or the inverse $r^{-}$ of a role name $r$. An ELI concept is formed according to the syntax rule $C, D ::= \top \mid A \mid C \sqcap D \mid \exists R . C$ where $A$ ranges over concept names and $R$ over roles. A basic concept $B$ is an ELI concept of the form $\top$, $A$, or $\exists R . T$. When dealing with basic concepts, for brevity we may write $\exists R$ in place of $\exists R . T$.

A DL-Lite$^H$ ontology $O$ is a finite set of concept inclusions (CIs) $B \subseteq C$, role inclusions (RIs) $R_1 \subseteq R_2$, concept disjointness constraints $B_1 \sqcap B_2 \subseteq \bot$, role disjointness constraints $R_1 \sqcap R_2 \subseteq \bot$, and functionality assertions $\text{func}(R)$. Here, $B, B_1,$ and $B_2$ range over basic concepts, $C$ over ELI concepts, and $R_1, R_2, R$ over roles. Superscript $\rightarrow$ indicates the presence of role inclusions (also called role hierarchies) and superscript $\leftrightarrow$ indicates functionality assertions, and thus it should be clear what we mean with a DL-Lite$^H$ ontology and with a DL-Lite$^F$ ontology. In fact, we are mainly interested in these two fragments of DL-Lite$^F$.

A DL-Lite$^H$ ontology is in normal form if all concept inclusions in it are of one of the forms $A \subseteq B$, $B \subseteq A$, and $A \subseteq \exists R . A'$ with $A, A'$ concept names or $\top$ and $B$ a basic concept. Note that CIs of the form $\exists R \subseteq \exists S$ are not admitted and neither are CIs of the form $A \subseteq \exists R . C$ with $C$ a compound concept. An ABox $A$ is a finite set of concept assertions $A(a)$ and role assertions $r(a, b)$ with $A$ a concept name or $\top$, $r$ a role name, and $a, b$ individual names. We use $\text{ind}(A)$ to denote the set of individual names used in $A$.

As usual, the semantics is given in terms of interpretations $I$, which we define to be a (possibly infinite and) non-empty set of concept and role assertions. We use $\Delta^I$ to denote the set of individual names in $I$, define $A^I = \{a \mid A(a) \in I\}$ for all $A \in N_C$, and $r^I = \{(a, b) \mid r(a, b) \in I\}$ and $(r^{-})^I = \{(b, a) \mid r(a, b) \in I\}$ for all $r \in N_R$. This definition of interpretation is slightly different from the usual one, but equivalent; its virtue is uniformity as every ABox is a finite interpretation. The interpretation function $\cdot^I$ can be extended from concept names to ELI concepts in the standard way [Baader et al., 2017]. An interpretation $I$ satisfies a concept or role inclusion $\alpha_1 \subseteq \alpha_2$ if $\alpha_1^I \subseteq \alpha_2^I$, a concept or role disjointness constraint $\alpha_1 \sqcap \alpha_2 \subseteq \bot$ if $\alpha_1^I \cap \alpha_2^I = \emptyset$, and a functionality assertion $\text{func}(R)$ if $R^I$ is a partial function. It satisfies a concept or role assertion $\alpha$ if $\alpha^I$. Note that, as usual, we thus make the standard names assumption, implying the unique name assumption.

An interpretation is a model of an ontology or an ABox if it satisfies all inclusions, disjointness constraints, and assertions in it. We write $O \models \alpha_1 \subseteq \alpha_2$ if every model of the ontology $O$ satisfies the concept or role inclusion $\alpha_1 \subseteq \alpha_2$ and $O \models \alpha_1 \equiv \alpha_2$ if $O \models \alpha_1 \subseteq \alpha_2$ and $O \models \alpha_2 \subseteq \alpha_1$. If $\alpha_1$ and $\alpha_2$ are basic concepts or roles, then such consequences are decidable in PTIME both in DL-Lite$^H$ and in DL-Lite$^F$ [Artale et al., 2009]. An ABox $A$ is satisfiable w.r.t. an ontology $O$ if $A$ and $O$ have a common model. Deciding ABox satisfiability is also in PTIME in both DL-Lite$^H$ and DL-Lite$^F$.

A signature is a set of concept and role names, uniformly referred to as symbols. For any syntactic object $O$ such as an ontology or an ABox, we use $\text{sig}(O)$ to denote the symbols $\top$.

1This depends on admitting assertions $\top(a)$ in ABoxes.
used in $O$ and $|O|$ to denote the size of $O$, that is, the length of a representation of $O$ as a word in a suitable alphabet.

**Queries.** An $EL$ concept $C$ can be viewed as an $EL$ query ($ELIQ$). An individual $a \in ind(A)$ is an answer to $C$ on an ABox $A$ w.r.t. an ontology $O$, written $A, O \models C(a)$, if $a \in C^*$ for all models $I$ of $A$ and $O$. We shall often view ELIQs as unary conjunctive queries (CQs) and also consider CQs that are not ELIQs. In this paper, CQs are always unary. A CQ thus takes the form $q(x_0) = \exists y \phi(x_0, y)$ with $\phi$ a conjunction of concept atoms $A(x)$ and role atoms $r(y, x)$ where $A \in NC$ and $r \in NR$. We use $\var(q)$ to denote the set of variables that occur in $q$. We may view $q$ as a set of atoms and may write $r^- (y, x)$ in place of $r(y, x)$. We call $x_0$ the answer variable and use the notion of an answer and the notation $A, O \models q(a)$ also for CQs. The formal definition is in terms of homomorphisms as usual, details are in the long version. ELIQs are in 1-to-1 correspondence with CQs. If a CQ $q$ is a set of atoms and may write $r^- (y, x)$ in place of $r(y, x)$. We call $x_0$ the answer variable and use the notion of an answer and the notation $A, O \models q(a)$ also for CQs. The formal definition is in terms of homomorphisms as usual, details are in the long version. ELIQs are in 1-to-1 correspondence with CQs.

**Definition 1.** A frontier of an ELIQ $q$ w.r.t. a DL-Lite$^H$ ontology $O$ is a set of ELIQs $F$ such that:

1. $q \subseteq q_F$ for all $q_F \in F$;
2. $q_F \not\subseteq q$ for all $q_F \in F$;
3. for all ELIQs $q'$ with $q \subseteq q' \not\subseteq q$, there is a $q_F \in F$ with $q_F \not\subseteq q'$.

It is not hard to see that finite frontiers that are minimal w.r.t. set inclusion are unique up to equivalence of the ELIQs in them, that is, if $F_1$ and $F_2$ are minimal frontiers of $q$ w.r.t. $O$, then for every $q_F \in F_1$ there is a $q'_F \in F_2$ such that $q_F \equiv q'_F$ and vice versa. The following is the main result of this section.

**Theorem 1.** Let $O$ be a DL-Lite$^H$ ontology and $q$ an ELIQ that is $O$-minimal and satisfiable w.r.t. $O$. Then a frontier of $q$ w.r.t. $O$ can be computed in polynomial time.

We note that Theorem 1 still holds when $O$-minimality is dropped as a precondition and Condition 2 of frontiers is dropped as well. For proving Theorem 1, we first observe that we can concentrate on ontologies that are in normal form.

**Lemma 1.** For every DL-Lite$^H$ ontology $O$, we can construct in polynomial time a DL-Lite$^H$ ontology $O'$ in normal form such that every $O$-minimal ELIQ $q$ is also $O'$-minimal and a frontier of $q$ w.r.t. $O$ can be constructed in polynomial time given a frontier of $q$ w.r.t. $O'$.

We now prove Theorem 1, adapting and generalizing a technique from [Ten Cate and Dalmia, 2021]. Let $O$ and $q(x_0)$ be as in the formulation of the theorem, with $O$ in normal form. We may assume w.l.o.g. that $q$ is $O$-saturated. To construct a frontier of $q$ w.r.t. $O$, we consider all ways to generalize $q$ in a least general way where ‘generalizing’ means to construct from $q$ an ELIQ $q'$ such that $q \subseteq q'$ and $q' \not\subseteq q$ and ‘least general way’ that there is no ELIQ $q$ that generalizes $q$ and satisfies $q \equiv q'$ and $q' \not\subseteq q$. We do this in two steps: the actual generalization plus a compensation step, the latter being needed to guarantee that we indeed arrive at a least general generalization.

For $x \in \var(q)$, we use $q_x$ to denote the ELIQ obtained from $q$ by taking the subtree of $q$ rooted at $x$ and making $x$ the answer variable. The construction that follows involves the introduction of fresh variables $x$, some of which are a ‘copy’ of a variable from $\var(q)$. We then use $x^+$ to denote that original variable.

**Step 1: Generalize.** For each variable $x \in \var(q)$, define a set $F_0(x)$ that contains all ELIQs which can be obtained by starting with $q_x(x)$ and then doing one of the following:

(A) **Drop concept atom:**
1. choose an atom $A(x) \in q$ such that
   (a) there is no $B(x) \in q$ with $O \models B \subseteq A$ and $O \not\models A \subseteq B$ and
   (b) there is no $R(x, y) \in q$ with $O \models \exists R \subseteq A$;
2. remove all $B(x) \in q$ with $O \models A \subseteq B$, including $A(x)$.
(B) **Generalize subquery:**
1. choose an atom $R(x, y) \in q$ directed away from $x_0$;
2. remove $R(x, y)$ and all atoms of $q_y$;
3. for each $q'(y) \in F_0(y)$, add a disjoint copy $q''$ of $q'$ and the role atom $R(x, y''')$ with $y'''$ the copy of $y$ in $q''$. 

2629
4. for every role \( S \) with \( \mathcal{O} \models R \sqsubseteq S \) and \( \mathcal{O} \models S \not\sqsubseteq R \), add a disjoint copy \( \tilde{q}_y \) of \( q_y \) and the role atom \( S(y, y') \) with \( y' \) the copy of \( y \) in \( \tilde{q}_y \).

The definition of \( x^i \) should be clear in all cases. In Point 3 of Case (B), for example, for every variable \( z \) in \( q' \) that was renamed to \( z' \) in \( \tilde{q}' \) set \( z'^i = z' \). Note that \( z'^i \) is defined for all variables \( z \) that occur in queries in \( \mathcal{F}_0(x) \). Also note that, in Point 1b of (A), it is important to use \( q \) rather than \( q_z \) as \( y \) could be a predecessor of \( x \) in \( q \).

**Step 2:** Compensate. We construct a frontier \( \mathcal{F} \) of \( q(x_0) \) by including, for each \( p \in \mathcal{F}_0(x_0) \), the ELIQ obtained from \( p \) by the following two steps. We write \( x \overset{R_{q, \mathcal{O}}}{\sim} A \) if \( A_q, \mathcal{O} \models \exists R.A(x) \) and there is no \( S(x, y) \in \mathcal{O} \) with \( S \subseteq R \) and \( A_q, \mathcal{O} \models A(y) \).

**Step 2A.** Consider all \( x \in \text{var}(p) \), roles \( R, S \), and concept names \( A \) such that \( x^i \overset{R_{q, \mathcal{O}}}{\sim} A \), \( \mathcal{O} \models R \sqsubseteq S \), and \( \mathcal{O} \models \exists S \subseteq B \) implies \( B(z) \in p \) for all concept names \( B \). Add the atoms \( S(x, z), A(z), R(x, z) \) where \( z, x' \) are fresh variables with \( z'^i \) undefined, \( x'^i = x^i \), and add a disjoint copy \( \tilde{q} \) of \( q \), glue the copy of \( x^i \) in \( \tilde{q} \) to \( x' \).

**Step 2B.** Consider every \( S(x, y) \in p \) directed away from \( x_0 \) that was not added in Step 2A. Then \( x^i \) and \( y^i \) are defined. For every role \( R \) with \( A_q, \mathcal{O} \models R(x^i, y^i) \), add an atom \( R(z, y), z \) a fresh variable with \( z^i = x^i \), as well as a disjoint copy \( \tilde{q} \) of \( q \) and glue the copy of \( x^i \) in \( \tilde{q} \) to \( z \).

This finishes the construction of the frontier of \( q \).

**Example 1.** Consider the DL-Lite\(^H\) ontology \( \mathcal{O} = \{ A \subseteq \exists r \subseteq A, r \subseteq s \} \) and the ELIQ \( q(x_0) = A(x_0) \land B(x_0) \).

Then \( \mathcal{F} \) contains the ELIQs \( p_1 \) and \( p_2 \) shown below:

\[
\begin{array}{c c c c c c}
A, B & x_1 & A, B & x_2 & A, B
\end{array}
\]

where \( \mathcal{F} p_1 \) is the result of dropping the concept atom \( A(x_0) \) and \( p_2 \) is the result of dropping the concept atom \( B(x_0) \). Step 2A adds an \( r \)-successor and an \( s \)-successor of \( x_0 \) in \( p_2 \) but only an \( s \)-successor in \( p_1 \) as \( \mathcal{O} \models \exists s \subseteq A \), and then attaches copies of \( q \). Step 2B does nothing, as all role atoms have been added in Step 2A.

**Example 2.** Consider the DL-Lite\(^H\) ontology \( \mathcal{O} = \{ r \subseteq s \} \) and the ELIQ \( q(x_0) \) shown on the left-hand side below:

\[
\begin{array}{c c c c}
x_0 & x_0 & x_0
\end{array}
\]

Then \( \mathcal{F} \) contains only the ELIQ \( p \) shown on the right-hand side. It is the result of dropping the concept atom \( A(y) \) in \( q_y \), then generalizing the subquery \( r(x_0, y) \) in \( q_{x_0} = q \), and then compensating. Step 2A of compensation adds nothing. Step 2B adds the two dashed role atoms and attaches copies of \( q \) to \( x_1 \) and \( x_2 \).

**Lemma 2.** \( \mathcal{F} \) is a frontier of \( q(x_0) \) w.r.t. \( \mathcal{O} \).

We next show that the constructed frontier is of polynomial size and that its computation takes only polynomial time.

**Lemma 3.** The construction of \( \mathcal{F} \) runs in time polynomial in \( ||q|| + ||\mathcal{O}|| \) (and thus \( \sum \mathcal{F} \mathcal{F} \) is polynomial in \( ||q|| + ||\mathcal{O}|| \)).

We next observe that adding conjunction to DL-Lite destroys polynomial frontiers and thus Theorem 1 does not apply to DL-Lite\(^\text{horn}\) ontologies [Artale et al., 2009]. In fact, this already holds for very simple queries and ontologies, implying that also for other DLs that support conjunction such as \( \mathcal{E} \mathcal{L} \), polynomial frontiers are elusive. A conjunction of atomic queries (AQ\(^+\)) is a unary CQ of the form \( q(x_0) = A_1(x_0) \land \cdots \land A_n(x_0) \) and a conjunctive ontology is a set of CIs of the form \( A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A \) where \( A_1, \ldots, A_n \) and \( A \) are concept names.

**Theorem 2.** There are families of AQ\(^+\)'s \( q_1, q_2, \ldots \) and conjunctive ontologies \( \mathcal{O}_1, \mathcal{O}_2, \ldots \) such that for all \( n \geq 1 \), any frontier of \( q_n \) w.r.t. \( \mathcal{O}_n \) has size at least \( 2^n \).

**4 Frontiers in DL-Lite\(^F\)**

We start by observing that frontiers of ELIQs w.r.t. DL-Lite\(^F\) ontologies may be infinite. This leads us to identifying a syntactic restriction on DL-Lite\(^F\) ontologies that regains finite frontiers. In fact, we show that they are of polynomial size and can be computed in polynomial time.

**Theorem 3.** There is an ELIQ \( q \) and a DL-Lite\(^F\) ontology \( \mathcal{O} \) such that \( q \) does not have a finite frontier w.r.t. \( \mathcal{O} \).

In the proof of Theorem 3, we use the ELIQ \( A(x) \) and \( \mathcal{O} = \{ A \subseteq \exists r, \exists^r \subseteq A, \exists r \subseteq s, \text{func}(r^-) \} \).

The universal model \( U_q, \mathcal{O} \) of \( A_q \) and \( \mathcal{O} \) is an infinite \( r \)-path on which every point has an \( s \)-successor. Now consider the following ELIQs \( q_1, q_2, \ldots \) that satisfy \( q_i \not\subseteq q \) for all \( i \geq 1 \):

\[
q_i(x_1) = r(x_1, x_2), \ldots, r(x_{n-1}, x_n), s(x_n), s(x_n', y), y, x_1', x_2', \ldots, r(x_{n-1}', x_n'), A(x_i')
\]

Any frontier \( \mathcal{F} \) must contain a \( p_i \) with \( p_i \subseteq q_i \) for all \( i \geq 1 \). We show that, consequently, there is no bound on the size of the queries in \( \mathcal{F} \). We invite the reader to apply the frontier construction from Section 3 after dropping \( \text{func}(r^-) \).

The proof actually shows that there is no finite frontier even if we admit the use of unrestricted CQs in the frontier in place of ELIQs. To regain finite frontiers, we restrict our attention to DL-Lite\(^F\) ontologies \( \mathcal{O} \) such that if \( B \subseteq C \) is a CI in \( \mathcal{O} \), then \( C \) contains no subconcepts of the form \( \exists R.D \) with \( \text{func}(R^-) \in \mathcal{O} \). We call such an ontology a DL-Lite\(^F\) ontology. We again concentrate on ontologies in normal form.

**Lemma 4.** For every DL-Lite\(^F\) ontology \( \mathcal{O} \), we can construct in polynomial time a DL-Lite\(^F\) ontology \( \mathcal{O}' \) in normal form such that for every ELIQ \( q \), a frontier of \( q \) w.r.t. \( \mathcal{O} \) can be constructed in polynomial time given a frontier of \( q \) w.r.t. \( \mathcal{O}' \).
The main result of this section is as follows.

**Theorem 4.** Let \( \mathcal{O} \) be a DL-Lite\(^{F^{-}} \) ontology and \( q \) an ELIQ that is satisfiable w.r.t. \( \mathcal{O} \). Then a frontier of \( q \) w.r.t. \( \mathcal{O} \) can be computed in polynomial time.

To prove Theorem 4, let \( \mathcal{O} \) and \( q \) be as in the theorem, \( \mathcal{O} \) in normal form. We may assume w.l.o.g. that \( q \) is \( \mathcal{O} \)-minimal and \( \mathcal{O} \)-saturated. The construction of a frontier follows the same general approach as for DL-Lite\(^{H} \), but the presence of functional roles significantly complicates the compensation step. As before, we introduce fresh variables and rely on the mapping \( x^k \).

**Step 1: Generalize.** For each variable \( x \in \text{var}(q) \), define a set \( F_0(x) \) that contains all ELIQs which can be obtained by starting with \( q_x(x) \) and then doing one of the following:

(A) Drop concept atom: exactly as for DL-Lite\(^{H} \).

(B) Generalize subquery:
1. choose an atom \( R(x, y) \in q \) directed away from \( x_0 \);
2. remove \( R(x, y) \) and all atoms of \( q_y \);
3. if func\((R) \notin \mathcal{O} \), then for each \( q'_y(y) \in F_0(y) \) add a disjoint copy \( q'_y \) of \( q'_y \) and the role atom \( R(x, y') \) with \( y' \) the copy of \( y \) in \( q'_y \);
4. if func\((R) \in \mathcal{O} \) and \( F_0(y) \neq \emptyset \), then choose and add a \( q'_y \in F_0(y) \) and the role atom \( R(x, y) \).

**Step 2: Compensate.** We construct a frontier \( \mathcal{F} \) of \( q(x_0) \) by including, for each \( p \in F_0(x_0) \), the CQ obtained from \( p \) by the following two steps. For \( x \in \text{var}(p) \), \( R \) a role, and \( M \) a set of concept names from \( \mathcal{O} \), we write \( x \xrightarrow{\text{R}}_{\mathcal{O}} M \) if \( M \) is maximal with \( A_q, \mathcal{O} \models \exists R. \bigwedge M(x) \) and there is no \( R(x, y) \in q \) with \( A_q, \mathcal{O} \models \bigwedge M(y) \).

**Step 2A.** Consider every \( x \in \text{var}(p) \), role \( R \), and set of concept names \( M = \{A_1, \ldots, A_k\} \) with \( x \xrightarrow{\text{R}}_{\mathcal{O}} M \). If \( \mathcal{O} \models \exists R \subseteq B \implies B(x) \in p \) for all concept names \( B \), add the atoms \( R(x, z), A_1(z), \ldots, A_k(z) \) where \( z \) is a fresh variable, and leave \( z^k \) undefined.

**Step 2B.** This step is iterative. For bookkeeping, we mark atoms \( R(x, y) \in p \) to be processed in the next round of the iteration. Marking is only applied to atoms \( R(x, y) \) directed away from \( x_0 \) such that \( y^k \) is defined and if \( x^k \) is undefined then func\((R^-) \notin \mathcal{O} \) or \( q \) contains no atom of the form \( R(y^k, z) \).

To start, consider every \( R(x, y) \in p \) directed away from \( x_0 \) with \( \text{func}(R^-) \notin \mathcal{O} \). Then \( x^k \) is defined. Extend \( p \) with atom \( R^-(y, x^k), x^k \) a fresh variable with \( x^k = x^k \). Mark the new atom.

Then repeatedly choose a marked atom \( R(x, y) \) and unmark it. If func\((R^-) \notin \mathcal{O} \) or \( q \) contains no atom of the form \( R(y^k, z) \), then add a disjoint copy \( \hat{q} \) of \( q \) and glue the copy of \( y^k \) in \( \hat{q} \) to \( y \). Otherwise, do the following:

(i) add \( A(y) \) whenever \( A_q, \mathcal{O} \models A(y) \);

(ii) for all atoms \( S(y^k, z) \in q \) and \( S(y^k, z) \neq R^-(y^k, x^k) \), extend \( p \) with atom \( S(y^k, z), z^k \) a fresh variable with \( z^k = z \). Mark \( S(y, z^k) \).

(iii) for all roles \( S \) and sets \( M = \{A_1, \ldots, A_k\} \) such that \( y^k \leadsto_{\mathcal{O}} S(y, u), S^-(u, y'), A_1(u), \ldots, A_k(u) \) where \( u \) and \( y' \) are fresh variables. Set \( y^k = y^k \) and mark \( S^-(u, y') \).

The step is repeated as long as possible. Note that in Point (iii), the role \( S \) must occur on the right-hand side of some CI in the DL-Lite\(^{F^{-}} \) ontology \( \mathcal{O} \). Consequently, func\((S^-) \notin \mathcal{O} \) and it is not a problem that \( u \) receives two \( S \)-predecessors. Also in Point (iii), func\((S) \in \mathcal{O} \) implies that \( q \) cannot contain an atom \( S(y^k, z) \) due to the definition of ‘\( \leadsto_{\mathcal{O}} \)’ and thus we may leave \( u^k \) undefined.

This finishes the construction of the frontier \( \mathcal{F} \) of \( q \).

**Example 3.** Consider the ontology \( \mathcal{O} = \{ \text{func}(s) \} \) and ELIQ \( q(x_0) \) shown on the left-hand side below:

- The ELIQ \( p \in \mathcal{F} \) shown on the right-hand side is the result of dropping the concept atom \( A(z) \) in \( q_{x_0} \), then generalizing the subquery \( s(x_0, z) \) in \( q_{x_0} = q \), and then compensating. Step 2A of compensation adds nothing. The start of Step 2B adds the two dashed role atoms and marks them. The step of Step 2B adds the dotted role atom via Point (ii) and marks it. When the step of Step 2B processes role atoms \( r^-(y, x_2) \) and \( r(x_0, y) \), it attaches copies of \( q \) to \( x_2 \) and \( y_1 \).

**Lemma 5.** \( \mathcal{F} \) is a frontier of \( q(x_0) \) w.r.t. \( \mathcal{O} \).

As for DL-Lite\(^{H} \), the constructed frontier is of polynomial size and its computation takes only polynomial time. Crucially, the iterative process in Point 2B terminates since in Step (ii) (a copy of a) subquery of \( q \) is added and the process stops at atoms added in Step (iii).

**Lemma 6.** The construction of \( \mathcal{F} \) runs in time polynomial in \( ||q|| + ||\mathcal{O}|| \) (and thus \( \sum_{p \in \mathcal{F}} ||p|| \) is polynomial in \( ||q|| + ||\mathcal{O}|| \)).

One first application of our results on frontiers is to the unique characterization of ELIQs w.r.t. ontologies by labeled data examples. Details are in the long version.

5 Learning ELIQs under Ontologies

We use our results on frontiers to show that ELIQs are polynomial-time learnable under ontologies formulated in DL-Lite\(^{H} \) and DL-Lite\(^{F^{-}} \), using only membership queries. We also present two results on non-learnability.

**Theorem 5.** ELIQs are polynomial time learnable under DL-Lite\(^{H} \) ontologies and under DL-Lite\(^{F^{-}} \) ontologies using only membership queries.

If the ontology contains concept disjointness constraints, then this only holds true if the learner is provided with a seed CQ (definition given below).

For proving Theorem 5, let \( \mathcal{O} \) be an ontology formulated in DL-Lite\(^{H} \) or DL-Lite\(^{F^{-}} \) and \( q_T(x_0) \) the target ELIQ known to the oracle. We may again assume \( \mathcal{O} \) to be in normal form.
Algorithm 1 Learning ELIQs under DL-Lite ontologies

Input An ontology $\mathcal{O}$ in normal form and a CQ $q^0_0$ satisifiable w.r.t. $\mathcal{O}$ such that $q^0_0 \subseteq \mathcal{O} q_T$

Output An ELIQ $q_H$ such that $q_H \equiv_\mathcal{O} q_T$

$q_H := \text{treeify}(q^0_0)$
while there is a $q_F \in \mathcal{F}_{q_H}$ with $q_F \subseteq \mathcal{O} q_R$ do
$q_H := \text{minimize}(q_F)$
end while
return $q_H$

Lemma 7. In DL-Lite$^H$ and DL-Lite$^F_-$, every polynomial time learning algorithm for ELIQs under ontologies in normal form that uses only membership queries can be transformed into a learning algorithm with the same properties for ELIQs under unrestricted ontologies.

The learning algorithm is displayed as Algorithm 1. It assumes a seed CQ $q^0_H$, that is, a CQ $q^0_0$ such that $q^0_0 \subseteq \mathcal{O} q_T$ and $q^0_0$ is satisfiable w.r.t. $\mathcal{O}$. If $\mathcal{O}$ contains no disjointness constraints, then for $\Sigma = \text{sig}(\mathcal{O})$ we can use as the seed CQ $q^0_0 = \{A(x_0) \mid A \in \Sigma \cap N_C \cup \{r(x_0, x_0) \mid r \in \Sigma \cap N_R\}$. We can still construct a seed CQ $q^0_0$ in time polynomial in $|\mathcal{O}|$ if $\mathcal{O}$ contains no disjoint constraints on concepts (but potentially on roles); details at in the long version. In the presence of concept disjointness constraints, a seed CQ can be obtained through an initial equivalence query.

The algorithm constructs and repeatedly updates a hypothesis ELIQ $q_H$ while maintaining the invariant $q_H \subseteq \mathcal{O} q_T$. The initial call to subroutine treeify yields an ELIQ $q_H$ with $q^0_H \subseteq \mathcal{O} q_H \subseteq \mathcal{O} q_T$ to be used as the first hypothesis. The algorithm then iteratively generalizes $q_H$ by constructing the frontier $\mathcal{F}_{q_H}$ of $q_H$ w.r.t. $\mathcal{O}$ in polynomial time and choosing from it a new ELIQ $q_H$ with $q_H \subseteq \mathcal{O} q_T$. In between, the algorithm applies the minimize subroutine to ensure that the new $q_H$ is $\mathcal{O}$-minimal and to avoid an excessive blowup while iterating in the while loop.

We next detail the subroutines treeify and minimize. We define minimize on unrestricted CQs since it is applied to non-ELIQs as part of the treeify subroutine.

The minimize subroutine. The subroutine takes as input a unary CQ $q(x_0)$ that is satisfiable w.r.t. $\mathcal{O}$ and satisfies $q \subseteq \mathcal{O} q_T$. It computes a unary CQ $q'$ with $q' \subseteq q' \subseteq \mathcal{O} q_T$ using membership queries that is minimal in a strong sense. Formally, minimize first makes sure that $q$ is $\mathcal{O}$-saturated and then exhaustively applies the following operation:

Remove atom. Choose a role atom $r(x, y) \in q$ and let $q^-$ be the maximal connected component of $q \setminus \{r(x, y)\}$ that contains $x_0$. Pose the membership query $\mathcal{A}_{q^-}$. $\mathcal{O} \models q_T(x_0)$. If the response is positive, continue with $q^-$ in place of $q$.

Clearly, the result of minimize is $\mathcal{O}$-minimal.

The treeify subroutine. The subroutine takes as input a unary CQ $q(x_0)$ that is satisfiable w.r.t. $\mathcal{O}$, and satisfies $q \subseteq \mathcal{O} q_T$. It computes an ELIQ $q'$ with $q' \subseteq q' \subseteq \mathcal{O} q_T$ by repeatedly increasing the length of cycles in $q$ and minimizing the obtained query; a similar construction is used in [ten Cate and Dalmau, 2021]. The resulting ELIQ is $\mathcal{O}$-minimal.

Formally, treeify first makes sure that $q(x_0)$ is $\mathcal{O}$-saturated and then constructs a sequence of CQs $p_1, p_2, \ldots$ starting with $p_1 = \text{minimize}(q)$ and then taking $p_{i+1} = \text{minimize}(p_i)$ where $p_i$ is obtained from $p_i$ by doubling the length of some cycle. Here, a cycle in a CQ $q$ is a sequence $R_1(x_1, x_2), \ldots, R_n(x_n, x_1)$ of distinct role atoms in $q$ such that $x_1, \ldots, x_n$ are distinct. More precisely, $p_i$ is the result of the following operation:

Double cycle. Choose a role atom $r(x, y) \in p_i$ that is part of a cycle in $p_i$ and let $p = p_i \setminus \{r(x, y)\}$. The CQ $p'$ is then obtained by starting with $p$, adding a disjoint copy $p'$ of $p$ where $x'$ refers to the copy of $x \in \text{var}(p)$ in $p'$ and adding the role atoms $r(x, y'), r(x', y)$.

If $p_i$ contains no more cycles, treeify stops and returns $p_i$.

6 Outlook

A natural next step for future work is to generalize the results presented in this paper to DL-Lite$^H_F$, adopting the same syntactic restriction that we have adopted for DL-Lite$^F$, and additionally requiring that functional roles have no proper subroles. The latter serves to control the interaction between functional roles and role inclusions. Even with this restriction, however, that interaction is very subtle and the frontier construction becomes significantly more complex. Other interesting questions are whether ELIQs can be learned in polynomial time w.r.t. DL-Lite$^H_\text{core}$ ontologies and whether CQs can be learned w.r.t. DL-Lite$^F_\text{core}$ ontologies when both membership and equivalence queries are admitted.

Acknowledgements

Carsten Lutz was supported by DFG CRC 1320 EASE.
References


