Journal of Machine Learning Research 19 (2018) 1-63 Submitted 5/16; Revised 2/18; Published 2/18

Exact Learning of Lightweight Description Logic Ontologies

Boris Konev KONEV@LIVERPOOL.AC.UK
Department of Computer Science
University of Liverpool, United Kingdom

Carsten Lutz CLU@INFORMATIK.UNI-BREMEN.DE
Department of Computer Science
University of Bremen, Germany

Ana Ozaki ANA.OZAKI@TU-DRESDEN.DE
Department of Computer Science
Dresden University of Technology, Germany

Frank Wolter WOLTER@LIVERPOOL.AC.UK
Department of Computer Science
University of Liverpool, United Kingdom

Editor: Luc De Raedt

Abstract

We study the problem of learning description logic (DL) ontologies in Angluin et al.'s framework of exact learning via queries. We admit membership queries ("is a given subsumption entailed by the target ontology?") and equivalence queries ("is a given ontology equivalent to the target ontology?"). We present three main results: (1) ontologies formulated in (two relevant versions of) the description logic DL-Lite can be learned with polynomially many queries of polynomial size; (2) this is not the case for ontologies formulated in the description logic $\mathcal{EL}$, even when only acyclic ontologies are admitted; and (3) ontologies formulated in a fragment of $\mathcal{EL}$ related to the web ontology language OWL 2 RL can be learned in polynomial time. We also show that neither membership nor equivalence queries alone are sufficient in cases (1) and (3).

Keywords: Exact Learning, Description Logic, Complexity

1. Introduction

In many subfields of artificial intelligence, ontologies are used to provide a common vocabulary for the application domain of interest and to give a meaning to the terms in the vocabulary, and to describe the relations between them. Description logics (DLs) are a prominent family of ontology languages with a long history that goes back to Brachman's famous knowledge representation system KL-ONE in the early 1980s (Brachman and Schmolze, 1985). Today, there are several widely used families of DLs that differ in expressive power, computational complexity, and intended application. The most important ones are the $\mathcal{ALC}$ family which aims at high expressive power, the $\mathcal{EL}$ family (Baader et al., 2005) which aims to provide scalable reasoning, and the DL-Lite family (Calvanese et al., 2007; Artale et al., 2009) which is tailored specifically towards applications in data access. In 2004, the World Wide Web Committee (W3C) has standardised a DL of the $\mathcal{ALC}$ family as an ontology language for the

©2018 Boris Konev, Carsten Lutz, Ana Ozaki and Frank Wolter.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v19/16-256.html.

web, called OWL. The standard was updated to OWL 2 in 2009, and since then comprises a family of five languages including the OWL 2 profiles OWL 2 EL, OWL 2 QL, and OWL 2 RL. While OWL 2 EL is based on $\mathcal{EL}$ and OWL 2 QL on DL-Lite, OWL 2 RL is closely related to the fragment of $\mathcal{EL}$ that is obtained by allowing only concept names on the right-hand side of concept inclusions. In this paper we study DLs from the $\mathcal{EL}$ and DL-Lite families. Designing an ontology for an application domain is a subtle, error-prone, and time consuming task. From its beginnings, DL research was driven by the aim to provide various forms of support for ontology engineers, assisting them in the design of high-quality ontologies; examples include the ubiquitous task of ontology classification (Baader et al., 2017), reasoning support for debugging ontologies (Wang et al., 2005; Schlobach et al., 2007), support for modular ontology design (Stuckenschmidt et al., 2009), and checking the completeness of the modelling in a systematic way (Baader et al., 2007). The same aim is pursued by the field of ontology learning, where the goal is to use machine learning techniques for various ontology engineering tasks such as to identify the relevant vocabulary of the application domain (Cimiano et al., 2010; Wong et al., 2012), to learn an initial version of the ontology that is then refined manually (Borchmann and Distel, 2011; Ma and Distel, 2013; Jiménez-Ruiz et al., 2015), and to learn concept expressions as building blocks of an ontology (Lehmann and Hitzler, 2010). For details we refer the reader to a collection of articles in ontology learning edited by Lehmann and Völker (2014) and Section 7.

In this paper we concentrate on learning the full logical structure of a description logic ontology. Our starting point is the observation that building a high-quality ontology relies on the successful communication between an ontology engineer and a domain expert because the former is typically not sufficiently familiar with the domain and the latter is rarely an expert in ontology engineering. We study the foundations of this communication process in terms of a simple communication model and analyse, within this model, the complexity of constructing a correct and complete domain ontology. Our model rests on the following assumptions:

1. The domain expert has perfect knowledge of the domain, but is not able to formalise or communicate the target ontology $\mathcal{O}$ to be constructed.

2. The domain expert is able to communicate the vocabulary (predicate symbols, which in the case of DLs take the form of concept and role names) of $\mathcal{O}$ and shares it with the ontology engineer. The ontology engineer knows nothing else about the domain.

3. The ontology engineer can pose queries to the domain expert which the domain expert answers truthfully. The main queries posed by the ontology engineer are of the form

   > "Is the concept inclusion $C \sqsubseteq D$ entailed by $\mathcal{O}$?"

4. In addition, the ontology engineer needs a way to find out whether the ontology $\mathcal{H}$ that has been constructed so far, called the hypothesis ontology, is complete. If not, he requests an example illustrating the incompleteness. The engineer can thus ask:

   > "Is the ontology $\mathcal{H}$ complete? If not, then return a concept inclusion $C \sqsubseteq D$ entailed by $\mathcal{O}$ but not by $\mathcal{H}$."

We are then interested in whether the target ontology $\mathcal{O}$ can be constructed with only polynomially many queries of polynomial size (polynomial query learnability) or, even better, with overall polynomial time (polynomial time learnability). In both cases, the polynomial is in the size of the ontology to be constructed plus the size of the counterexamples returned by the domain expert. Without taking into account the latter, one can never expect to achieve polynomial time learnability because the domain expert could provide unnecessarily large counterexamples. Note that polynomial time learnability implies polynomial query learnability, but that the converse is false because polynomial query learnability allows the ontology engineer to run computationally costly procedures between posing queries.

The above model is an instance of Angluin et al.'s framework of exact learning via queries (Angluin, 1987b). In this context, the queries mentioned in Point 3 above are called *membership queries*. The queries in Point 4 are a form of *equivalence queries*. In Angluin's framework, however, such queries are slightly more general:

> "Is the hypothesis ontology $\mathcal{H}$ equivalent to the target ontology $\mathcal{O}$? If not, then return a concept inclusion $C \sqsubseteq D$ entailed by $\mathcal{O}$ but not by $\mathcal{H}$ (a *positive counterexample*) or vice versa (a *negative counterexample*)."

In our upper bounds (that is, polynomial learnability results), we admit only queries of the more restricted form in Point 4 above: the learning algorithm is designed in a way so that the hypothesis ontology $\mathcal{H}$ is a consequence of the target ontology $\mathcal{O}$ at all times, and thus the only meaningful equivalence query is a query of the form "Is $\mathcal{H}$ already complete?". Our lower bounds (results saying that polynomial learnability is impossible), in contrast, apply to unrestricted equivalence queries, that is, they do not assume that the hypothesis is implied by the target. In this way, we achieve maximum generality.

Within the setup outlined above, we study the following description logics:

(a) DL-Lite$^{\exists}_{\mathcal{R}}$, which is a member of the DL-Lite family that admits role inclusions and allows nested existential quantification on the right-hand side of concept inclusions;

(b) the extension DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ of DL-Lite$^{\exists}_{\mathcal{R}}$ with conjunction on the left-hand side of concept inclusions;

(c) the basic member $\mathcal{EL}$ of the $\mathcal{EL}$ family;

(d) the fragment $\mathcal{EL}_{\mathsf{lhs}}$ of $\mathcal{EL}$ where only concept names (but no compound concept expressions) are admitted on the right-hand side of concept inclusions.

We remark that DL-Lite$^{\exists}_{\mathcal{R}}$ is closely related to OWL 2 QL, which is based on the fragment of DL-Lite$^{\exists}_{\mathcal{R}}$ that does not allow nested existential quantification on the right-hand side of concept inclusions. In this more restricted case, though, polynomial learnability is uninteresting. In fact, the number of concept inclusions formulated in a fixed finite vocabulary Σ is bounded polynomially in the size of Σ instead of being infinite as in the description logics studied in this paper; consequently, TBoxes are trivially learnable in polynomial time, even when only membership queries (but no equivalence queries) are available or vice versa. The extension DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ of DL-Lite$^{\exists}_{\mathcal{R}}$ is not part of the OWL 2 QL standard, but admitting conjunctions on the left-hand side of concept inclusions is a useful and widely considered extension of basic DL-Lite dialects (Artale et al., 2009). $\mathcal{EL}_{\mathsf{lhs}}$ is a significant part

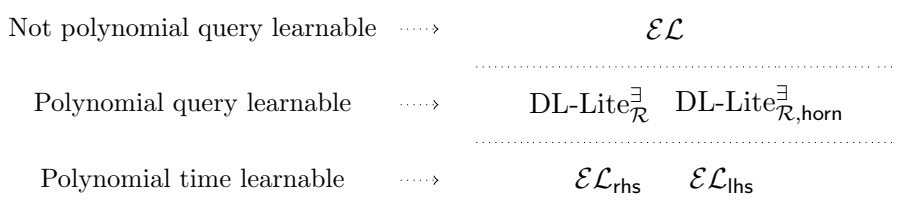

Figure 1: Summary of main results

of the OWL 2 RL language and can be viewed as a natural fragment of Datalog. An even better approximation of OWL 2 RL would be the extension of $\mathcal{EL}_{\mathsf{lhs}}$ with inverse roles, but polynomial learnability in that language remains an open problem. And finally, unrestricted $\mathcal{EL}$ can be viewed as a logical core of the OWL 2 EL language.

After introducing preliminaries in Section 2, we study exact learning of DL-Lite$^{\exists}_{\mathcal{R}}$ ontologies in Section 3, establishing polynomial query learnability. We strengthen this result to DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ in Section 4, using a significantly more subtle algorithm. It remains open whether DL-Lite$^{\exists}_{\mathcal{R}}$ and DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ admit polynomial time learnability. Our algorithms do not yield such a stronger result since they use subsumption checks to analyse counterexamples provided by the oracle and to integrate them into the current hypothesis ontology, and subsumption is NP-complete in these DLs (Kikot et al., 2011). In Section 5, we show that $\mathcal{EL}_{\mathsf{lhs}}$ ontologies are learnable in polynomial time, a result that extends the known polynomial time learnability of propositional Horn formulas (Angluin et al., 1992), which correspond to $\mathcal{EL}$ ontologies without existential restrictions. In fact, our algorithms take inspiration from learning algorithms for propositional Horn formulas and combine the underlying ideas with modern concepts from DL such as canonical models, simulations, and products. The algorithm for $\mathcal{EL}_{\mathsf{lhs}}$ also uses subsumption checks, which in this case does not get in the way of polynomial time learnability since subsumption in $\mathcal{EL}_{\mathsf{lhs}}$ can be decided in polynomial time.

In Section 6, we then establish that $\mathcal{EL}$ ontologies are not polynomial query learnable. Note that the fragment $\mathcal{EL}_{\mathsf{rhs}}$ of $\mathcal{EL}$, which is symmetric to $\mathcal{EL}_{\mathsf{lhs}}$ and only admits concept names on the *left*-hand side of concept inclusions is a fragment of DL-Lite$^{\exists}_{\mathcal{R}}$. Together, our upper bounds for DL-Lite$^{\exists}_{\mathcal{R}}$ and $\mathcal{EL}_{\mathsf{lhs}}$ thus establish that failure of polynomial query learnability of $\mathcal{EL}$ ontologies is caused by the interaction between existential restrictions on the left- and right-hand sides of concept inclusions. Interestingly, our result already applies to acyclic $\mathcal{EL}$ TBoxes, which disallow recursive definitions of concepts and are of a rather restricted syntactic form. However, the result does rely on concept inclusions as counterexamples that are of a form not allowed in acyclic TBoxes. We also show that ontologies formulated in DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ and in $\mathcal{EL}_{\mathsf{lhs}}$ are neither polynomial query learnable with membership queries alone nor with equivalence queries alone; corresponding results for propositional Horn formulas are well known (Frazier and Pitt, 1993; Angluin et al., 1992; Angluin, 1987a; Arias and Balcázar, 2011). Figure 1 summarises the main results obtained in this paper.

In Section 7 we provide an extensive discussion of related work on the exact learning of logical formulas and theories, and we close the paper with a discussion of open problems. A small number of proofs are deferred to an appendix.

2. Preliminaries

We introduce the description logics studied in this paper, then consider a representation of concept expressions in terms of labelled trees and show how important semantic notions such as subsumption between concept expressions can be characterised by homomorphisms between the corresponding trees. This also involves introducing canonical models, which are an important tool throughout the paper. Finally, we formally introduce the framework of exact learning.

2.1 Description Logics

Let $\mathsf{N_C}$ be a countably infinite set of *concept names* (denoted by upper case letters A, B, etc) and let $\mathsf{N_R}$ be a countably infinite set of *role names* disjoint from $\mathsf{N_C}$ (denoted by lower case letters r, s, etc). Concept and role names can be regarded as unary and binary predicates, respectively. In description logic, constructors are used to define compound concept and role expressions from concept and role names. In this paper, the only role constructor is the inverse role constructor: for $r \in \mathsf{N_R}$, the expression r^- is the *inverse role* of r. Semantically, r^- represents the converse of the binary relation r. A *role expression* is a role name or an inverse role. We set $r^- := s$ if $r = s^-$ for a role name s. For brevity, we will typically speak of *roles* rather than of role expressions. The concept constructors used in this paper are $\top$ (*everything*), $\sqcap$ (*conjunction*), and $\exists r.C$ (*qualified existential restriction*). Formally, *concept expressions* C are defined according to the following syntactic rule:

$$C, D \quad := \quad \top \quad | \quad A \quad | \quad C \sqcap D \quad | \quad \exists r.C$$

where A is a concept name and r is a role. For example, $\exists \mathsf{hasChild}.\top \sqcap \exists \mathsf{gender}.\mathsf{Male}$ denotes the class of individuals who have a child and whose gender is male.

Terminological knowledge is captured by finite sets of *inclusions* between concept expressions or roles. Specifically,

- a *concept inclusion (CI)* takes the form $C \sqsubseteq D$, where C and D are concept expressions, and
- a *role inclusion (RI)* takes the form $r \sqsubseteq s$, where r and s are roles.

An *ontology* or *TBox* is a finite set of CIs and RIs.[1] We use $C \equiv D$ as an abbreviations for the two CIs $C \sqsubseteq D$ and $D \sqsubseteq C$ and likewise for $r \equiv s$; we speak of *concept equivalences (CEs)* and *role equivalences (REs)*, respectively.

1. In the description logic literature, CIs of the form introduced here are often called $\mathcal{ELI}$ CIs to distinguish them from CIs that use concept expressions formulated in other description logics. The TBoxes are called $\mathcal{ELIH}$ TBoxes (TBoxes that consist of $\mathcal{ELI}$ CIs and RIs).

Example 1 *Consider the following TBox:*

$$\begin{aligned}
\mathsf{Prof} &\sqsubseteq \exists \mathsf{supervisor_of}.\mathsf{Student} \sqcap \exists \mathsf{conduct_research}.\top && (1)\\
\mathsf{Graduate} &\equiv \exists \mathsf{has_degree}.\top && (2)\\
\mathsf{GraduateStudent} &\equiv \mathsf{Student} \sqcap \mathsf{Graduate} && (3)\\
\mathsf{GraduateStudent} &\sqsubseteq \exists \mathsf{supervisor_of}^{-}.\mathsf{Prof} && (4)\\
\mathsf{supervisor_of} &\sqsubseteq \mathsf{advisor_of} && (5)\\
\mathsf{CS_Graduate} &\equiv \exists \mathsf{has_degree}.\mathsf{CS_Degree} && (6)
\end{aligned}$$

The CI in Line 1 states that every professor supervises students and conducts research. Notice that we do not specify the specific area of research, hence we use an unqualified existential restriction of the form $\exists r.\top$. The CE in Line 2 defines a graduate as anyone who has a degree. The CE in Line 3 defines a graduate student as a student who is a graduate. The CI in Line 4 states that graduate students are supervised by professors. Notice that we use the inverse role of supervisor_of *here. Line 5 shows an RI which states that every supervisor is an advisor. The CE in the last line defines a computer science graduate as someone with a degree in computer science.*

A *signature* is a set of concept and role names and we use $\Sigma_{\mathcal{T}}$ to denote the signature of the TBox $\mathcal{T}$, that is, the set of concept and role names that occur in it. The *size* $|C|$ of a concept expression C is the length of the string that represents C, where concept names and role names are considered to be of length one. The *size* $|\mathcal{T}|$ of a TBox $\mathcal{T}$ is defined as $\sum_{C \sqsubseteq D \in \mathcal{T}} |C| + |D|$.

The semantics of concept expressions and TBoxes is defined as follows (Baader et al., 2017). An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is given by a non-empty set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a mapping $\cdot^{\mathcal{I}}$ that maps every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and every role name r to a subset $r^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The *interpretation* $r^{\mathcal{I}}$ of an inverse role $r = s^{-}$ is given by $r^{\mathcal{I}} = \{(d, d') \mid (d', d) \in s^{\mathcal{I}}\}$ and the *interpretation* $C^{\mathcal{I}}$ of a concept expression C is defined inductively by

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}}\\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}\\
(\exists r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{there exists } d' \in C^{\mathcal{I}} \text{ with } (d, d') \in r^{\mathcal{I}}\}.
\end{aligned}$$

An interpretation $\mathcal{I}$ *satisfies* a concept expression C if $C^{\mathcal{I}}$ is not empty. It satisfies the CI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, written as $\mathcal{I} \models C \sqsubseteq D$. Similarly, $\mathcal{I}$ satisfies RI $r \sqsubseteq s$ if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$, written as $\mathcal{I} \models r \sqsubseteq s$. $\mathcal{I}$ is a *model of a TBox* $\mathcal{T}$ if it satisfies all CIs and RIs in $\mathcal{T}$. A TBox $\mathcal{T}$ *entails* a CI or RI α, in symbols $\mathcal{T} \models \alpha$, if α is satisfied in every model of $\mathcal{T}$. Concept expressions C and D are *equivalent w.r.t.* $\mathcal{T}$, written $\mathcal{T} \models C \equiv D$, if $\mathcal{T} \models C \sqsubseteq D$ and $\mathcal{T} \models D \sqsubseteq C$; equivalence of roles r and s is defined accordingly, written $\mathcal{T} \models s \equiv r$. TBoxes $\mathcal{T}$ and $\mathcal{T}'$ are *logically equivalent*, in symbols $\mathcal{T} \equiv \mathcal{T}'$, if $\mathcal{T} \models \alpha$ for all $\alpha \in \mathcal{T}'$ and vice versa.

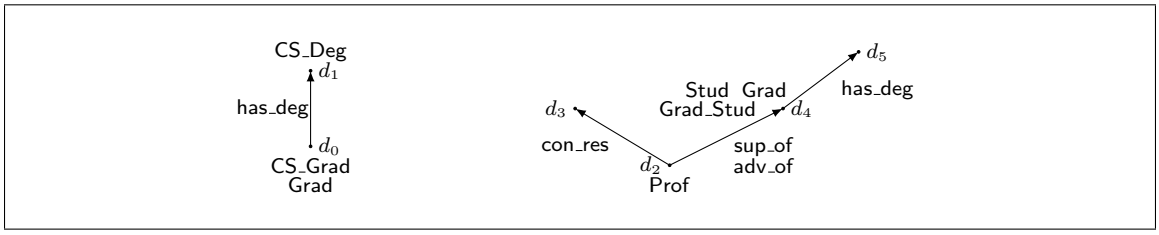

Figure 2: Illustration to Example 2.

Example 2 *Consider the TBox $\mathcal{T}$ from Example 1 and the interpretation $\mathcal{I}$ that is illustrated in Figure 2 and defined by setting $\Delta^{\mathcal{I}} = \{d_0, \ldots, d_5\}$ and*

$$\begin{aligned}
\mathsf{Prof}^{\mathcal{I}} &= \{d_2\}, & \mathsf{conduct_research}^{\mathcal{I}} &= \{(d_2, d_3)\}\\
\mathsf{Student}^{\mathcal{I}} &= \{d_4\}, & \mathsf{supervisor_of}^{\mathcal{I}} &= \{(d_2, d_4)\},\\
\mathsf{Graduate_Student}^{\mathcal{I}} &= \{d_4\}, & \mathsf{advisor_of}^{\mathcal{I}} &= \{(d_2, d_4)\},\\
\mathsf{Graduate}^{\mathcal{I}} &= \{d_4, d_0\}, & \mathsf{has_degree}^{\mathcal{I}} &= \{(d_4, d_5), (d_0, d_1)\},\\
\mathsf{CS_Graduate}^{\mathcal{I}} &= \{d_0\}, & \mathsf{CS_Degree}^{\mathcal{I}} &= \{d_1\}.
\end{aligned}$$

It is easy to see that $\mathcal{I}$ is a model of $\mathcal{T}$. Moreover, $\mathcal{I} \not\models \mathsf{Graduate} \sqsubseteq \mathsf{CS\_Graduate}$ as $\mathsf{Graduate}^{\mathcal{I}} = \{d_4, d_0\}$ but $\mathsf{CS\_Graduate}^{\mathcal{I}} = \{d_0\}$, thus $\mathcal{T} \not\models \mathsf{Graduate} \sqsubseteq \mathsf{CS\_Graduate}$. It can be shown that $\mathcal{T} \models \mathsf{CS\_Graduate} \sqsubseteq \mathsf{Graduate}$.

It is ExpTime-complete to decide, given a TBox $\mathcal{T}$ and a concept inclusion $C \sqsubseteq D$, whether $\mathcal{T} \models C \sqsubseteq D$ (Baader et al., 2008); this reasoning problem is known as *subsumption*. Because of this high complexity, the profiles of OWL 2 are based on syntactically more restricted description logics in which subsumption is less complex. We next introduce a few relevant such logics. A *basic concept* is a concept name or a concept expression of the form $\exists r.\top$, where r is a role. For example, $\exists \mathsf{hasChild}^-.\top$ is a basic concept, but $\exists \mathsf{hasChild}^-.\mathsf{Graduate}$ is not.

DL-Lite$^{\exists}_{\mathcal{R}}$. A *DL-Lite$^{\exists}_{\mathcal{R}}$ CI* takes the form

$$B \sqsubseteq C$$

where B is a basic concept and C is a concept expression. A *DL-Lite$^{\exists}_{\mathcal{R}}$ inclusion* is a DL-Lite$^{\exists}_{\mathcal{R}}$ CI or an RI. A *DL-Lite$^{\exists}_{\mathcal{R}}$ TBox* is a finite set of DL-Lite$^{\exists}_{\mathcal{R}}$ inclusions.

Example 3 *Lines* (1), (4), *and* (5) *of Example 1 are DL-Lite$^{\exists}_{\mathcal{R}}$ inclusions and Line* (2) *abbreviates the two DL-Lite$^{\exists}_{\mathcal{R}}$ CIs* $\mathsf{Graduate} \sqsubseteq \mathsf{has\_degree}.\top$ *and* $\mathsf{has\_degree}.\top \sqsubseteq \mathsf{Graduate}$. *Lines* (3) *and* (6) *do not fall within DL-Lite$^{\exists}_{\mathcal{R}}$.*

DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$. In the extension DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ of DL-Lite$^{\exists}_{\mathcal{R}}$, CIs take the form

$$B_1 \sqcap \cdots \sqcap B_n \sqsubseteq C$$

where $B_1, \ldots, B_n$ are basic concepts and C is a concept expression. A *DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBox* $\mathcal{T}$ is a finite set of DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ CIs and RIs. Both DL-Lite$^{\exists}_{\mathcal{R}}$ and DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ have been investigated in detail (Artale et al., 2009).

Example 4 *As Lines* (1), (2), (4) *and* (5) *from Example 1 fall within DL-Lite*$^{\exists}_{\mathcal{R}}$*, they also fall within DL-Lite*$^{\exists}_{\mathcal{R},\mathsf{horn}}$*. Line* (3) *falls within DL-Lite*$^{\exists}_{\mathcal{R},\mathsf{horn}}$*. Line* (6) *is not in DL-Lite*$^{\exists}_{\mathcal{R},\mathsf{horn}}$*.*

$\mathcal{EL}$. An $\mathcal{EL}$ *concept expression* is a concept expression that does not use inverse roles. An $\mathcal{EL}$ *concept inclusion* is a CI of the form

$$C \sqsubseteq D$$

where C and D are $\mathcal{EL}$ concept expressions. An $\mathcal{EL}$ *TBox* is a finite set of $\mathcal{EL}$ CIs. Thus, $\mathcal{EL}$ does neither admit role inclusions nor inverse roles. In contrast to DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$, however, it allows existential restrictions $\exists r.C$ with $C \neq \top$ on the left-hand side of CIs.

Example 5 *Inclusions* (1), (2), (3) *and* (6) *from Example 1 are $\mathcal{EL}$ inclusions. Inclusion* (5) *is not an $\mathcal{EL}$ inclusion.*

Subsumption is NP-complete in DL-Lite$^{\exists}_{\mathcal{R}}$ and in DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ (Kikot et al., 2011; Calvanese et al., 2007; Artale et al., 2009). Subsumption in $\mathcal{EL}$ is in PTime (Baader et al., 2005) and this is still true if RIs that do not use inverse roles are admitted in the TBox. Given a TBox $\mathcal{T}$ and an RI $r \sqsubseteq s$, deciding whether $\mathcal{T} \models r \sqsubseteq s$ is possible in PTime in all description logics considered in this paper. In fact, $\mathcal{T} \models r \sqsubseteq s$ if, and only if, there exists a sequence $r_0, \dots, r_n$ of roles such that $r = r_0$, $s = r_n$, and for every $i < n$ either $r_i \sqsubseteq r_{i+1} \in \mathcal{T}$ or $r_i^- \sqsubseteq r_{i+1}^- \in \mathcal{T}$. Our learning algorithms will carry out various subsumption checks as a subprocedure, as detailed later on.

2.2 Tree Representation of Concept Expressions

To achieve an elegant and succinct exposition of our learning algorithms, it will be convenient to represent concept expressions C as a finite directed tree T_C whose nodes are labelled with sets of concept names and whose edges are labelled with roles, and to describe manipulations of concept expressions in terms of manipulations of the corresponding tree such as merging nodes, replacing subgraphs, modifying node and edge labels, etc. We generally use ρ_C to denote the root node of the tree T_C. In detail, T_C is defined as follows. For $C = \top$, the tree T_C has a single node d with label $l(d) = \emptyset$; if $C = A$, where A is a concept name, then T_C has a single node d with $l(d) = \{A\}$; if $C = \exists r.D$, then T_C is obtained from T_D by adding a new root d_0 and an edge from d_0 to the root d of T_D with label $l(d_0, d) = r$ (we then call d an *r-successor* of d_0); if $C = D_1 \sqcap D_2$, then T_C is obtained by identifying the roots of T_{D_1} and T_{D_2}.

Example 6 *For $C = \mathsf{Student} \sqcap \exists \mathsf{has\_degree}.\exists \mathsf{has\_degree}^-.\mathsf{Graduate\_Student}$, T_C has three nodes, e_0, e_1, e_2, where e_0 is the root ρ_C of T_C, e_1 is a successor of e_0 and e_2 is a successor of e_1, the labelling of the nodes is given by $l(e_0) = \{\mathsf{Student}\}$, $l(e_1) = \emptyset$, and $l(e_2) = \{\mathsf{Graduate\_Student}\}$, and the labelling of the edges is given by $l(e_0, e_1) = \mathsf{has\_degree}$ and $l(e_1, e_2) = \mathsf{has\_degree}^-$, see Figure 3 (left).*

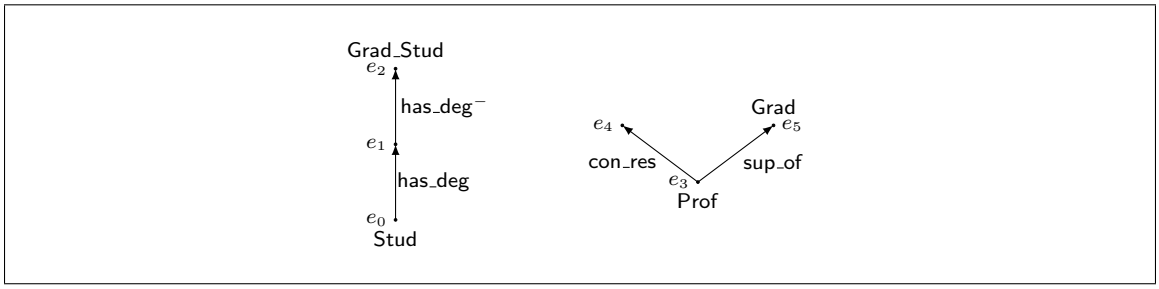

Figure 3: Illustration to Examples 6 (left) and 7 (right).

Conversely, every labelled finite directed tree T of the described form gives rise to a concept expression C_T in the following way: if T has a single node d labelled by $\{A_1, \ldots, A_n\}$, then $C_T = A_1 \sqcap \cdots \sqcap A_n$ (we treat $\top$ as the empty conjunction here, so if $l(d) = \emptyset$ then $C_T = \top$). Inductively, let d be the root of T labelled with $l(d) = \{A_1, \ldots, A_n\}$, let $d_1, \ldots, d_m$ be the successors of d, and let $l(d, d_1) = r_1, \ldots, l(d, d_m) = r_m$. Assume $C_{d_1}, \ldots, C_{d_m}$ are the concept expressions corresponding to the subtrees of T with roots $d_1, \ldots, d_m$, respectively. Then $C_T = A_1 \sqcap \cdots \sqcap A_n \sqcap \exists r_1.C_{d_1} \sqcap \cdots \sqcap \exists r_m.C_{d_m}$.

Example 7 *Let T be the tree with root e_3 labelled by $\{\mathsf{Prof}\}$ and successors e_4, e_5 labelled by $\emptyset$ and $\{\mathsf{Graduate}\}$, respectively, and with edge labelling given by $l(e_3, e_4) = \mathsf{conduct\_research}$ and $l(e_3, e_5) = \mathsf{supervisor\_of}$. Then*

$$C_T = \mathsf{Prof} \sqcap \exists \mathsf{conduct\_research}.\top \sqcap \exists \mathsf{supervisor\_of}.\mathsf{Graduate};$$

see Figure 3 (right).

In what follows, we will not always distinguish explicitly between C and its tree representation T_C which allows us to speak, for example, about the nodes and subtrees of a concept expression.

One important use of the tree representation of concept expressions is that both the truth relation '$d \in C^{\mathcal{I}}$' and the entailment '$\mathcal{T} \models C \sqsubseteq D$' can be characterised in terms of homomorphisms between labelled trees and interpretations. A mapping h from a tree T_C corresponding to a concept expression C to an interpretation $\mathcal{I}$ is a *homomorphism* if $A \in l(d)$ implies $h(d) \in A^{\mathcal{I}}$ for every concept name A and $r = l(d, d')$ implies $(h(d), h(d')) \in r^{\mathcal{I}}$ for all role names r. The following characterisation of the truth relation $d \in C^{\mathcal{I}}$ by means of homomorphisms is well-known.

Lemma 8 *Let $\mathcal{I}$ be an interpretation, $d \in \Delta^{\mathcal{I}}$, and C a concept expression. Then $d \in C^{\mathcal{I}}$ if, and only if, there is a homomorphism from T_C to $\mathcal{I}$ mapping ρ_C to d.*

The proof is by a straightforward induction on the structure of C (Baader et al., 1999).

Example 9 *Consider the interpretation $\mathcal{I}$ from Example 2 and the tree representations of the concept expressions given in Figure 3. It can be seen that functions g and h defined as $g(e_0) = d_4$, $g(e_1) = d_5$, $g(e_2) = d_4$ and $h(e_3) = d_2$, $h(e_4) = d_3$, $h(e_5) = d_4$ are homomorphisms and so, by Lemma 8, $d_4 \in (\mathsf{Student} \sqcap \exists \mathsf{has\_degree}.\exists \mathsf{has\_degree}^-.\mathsf{Graduate\_Student})^{\mathcal{I}}$ and $d_2 \in (\mathsf{Prof} \sqcap \exists \mathsf{conduct\_research}.\top \sqcap \exists \mathsf{supervisor\_of}.\mathsf{Graduate})^{\mathcal{I}}$.*

It is also standard to characterise the subsumption relation $\emptyset \models C \sqsubseteq D$ (that is, subsumption relative to the empty TBox) by means of homomorphisms between the tree representations T_D and T_C. A *homomorphism* h from labelled tree T_1 to labelled tree T_2 is a mapping from the nodes of T_1 to the nodes of T_2 such that $A \in l(d)$ implies $A \in l(h(d))$ for every concept name A and $r = l(d, d')$ implies $r = l(h(d), h(d'))$ for every role r.

Lemma 10 *Let C and D be concept expressions. Then $\emptyset \models C \sqsubseteq D$ if, and only if, there is a homomorphism from T_D to T_C that maps ρ_D to ρ_C.*

The 'if' direction is essentially a consequence of Lemma 8 and the fact that the composition of two homomorphisms is again a homomorphism. For the 'only if' direction, one can consider T_C as an interpretation $\mathcal{I}$ and apply Lemma 8 (Baader et al., 1999).

Next, we characterise subsumption in the presence of TBoxes in terms of homomorphisms. To achieve this, we make use of the *canonical model* $\mathcal{I}_{C_0,\mathcal{T}}$ of a concept expression C_0 and a TBox $\mathcal{T}$. If $\mathcal{T} = \emptyset$, then we want $\mathcal{I}_{C_0,\mathcal{T}}$ to be T_{C_0} viewed as a tree-shaped interpretation which we denote by $\mathcal{I}_{C_0}$ rather than by $\mathcal{I}_{C_0,\mathcal{T}}$. More precisely, the domain of $\mathcal{I}_{C_0}$ is the set of nodes of T_{C_0} and

$$\begin{array}{rll} d \in A^{\mathcal{I}_{C_0}} & \text{iff} & A \in l(d), \text{ for all } d \in \Delta^{\mathcal{I}_{C_0}} \text{ and concept names } A \\ (d, d') \in r^{\mathcal{I}_{C_0}} & \text{iff} & r = l(d, d'), \text{ for all } d, d' \in \Delta^{\mathcal{I}_{C_0}} \text{ and roles names } r \end{array}$$

We call the root ρ_{C_0} of T_{C_0} the *root of* $\mathcal{I}_{C_0}$. If $\mathcal{T} \neq \emptyset$, then $\mathcal{I}_{C_0,\mathcal{T}}$ is obtained by extending $\mathcal{I}_{C_0}$ so that the CIs in $\mathcal{T}$ are satisfied. For example, if $\mathcal{T} = \{A \sqsubseteq \exists r.B\}$ and $C_0 = A$, then $\mathcal{I}_{C_0}$ is a single node ρ_{C_0} with $A^{\mathcal{I}_{C_0}} = \{\rho_{C_0}\}$ and $X^{\mathcal{I}_{C_0}} = \emptyset$ for all concept and role names X distinct from A. To define $\mathcal{I}_{C_0,\mathcal{T}}$ we add a node d to $\Delta^{\mathcal{I}_{C_0}}$ and set $B^{\mathcal{I}_{C_0,\mathcal{T}}} = \{d\}$ and $r^{\mathcal{I}_{C_0,\mathcal{T}}} = \{(\rho_{C_0}, d)\}$. In general, $\mathcal{I}_{C_0,\mathcal{T}}$ is defined as the limit of a sequence $\mathcal{I}_0, \mathcal{I}_1, \ldots$ of interpretations, where $\mathcal{I}_0 = \mathcal{I}_{C_0}$. For the inductive definition of the sequence, assume that $\mathcal{I}_n$ has been defined. Then obtain $\mathcal{I}_{n+1}$ by applying one of the following rules once:

1. if $C \sqsubseteq D \in \mathcal{T}$ and $d \in C^{\mathcal{I}_n}$ but $d \notin D^{\mathcal{I}_n}$, then take the interpretation $\mathcal{I}_D$ and add it to $\mathcal{I}_n$ by identifying its root ρ_C with d. In more detail, assume that $\Delta^{\mathcal{I}_n} \cap \Delta^{\mathcal{I}_C} = \{d\}$ and $d = \rho_C$ and define $\mathcal{I}_{n+1}$ by setting, for all concept names A and role names r:

$$\Delta^{\mathcal{I}_{n+1}} = \Delta^{\mathcal{I}_n} \cup \Delta^{\mathcal{I}_C}, \quad A^{\mathcal{I}_{n+1}} = A^{\mathcal{I}_n} \cup A^{\mathcal{I}_C}, \quad r^{\mathcal{I}_{n+1}} = r^{\mathcal{I}_n} \cup r^{\mathcal{I}_C};$$

2. if $r \sqsubseteq s \in \mathcal{T}$ and $(d, d') \in r^{\mathcal{I}_n}$ but $(d, d') \notin s^{\mathcal{I}_n}$, then define $\mathcal{I}_{n+1}$ as $\mathcal{I}_n$ except that $s^{\mathcal{I}_{n+1}} := s^{\mathcal{I}_n} \cup \{(d, d')\}$ if s is a role name; otherwise there is a role name s_0 with $s = s_0^-$ and we define $\mathcal{I}_{n+1}$ as $\mathcal{I}_n$ except that $s_0^{\mathcal{I}_{n+1}} = s_0^{\mathcal{I}_n} \cup \{(d', d)\}$.

We assume that rule application is fair, that is, if a rule is applicable in a certain place, then it will indeed eventually be applied there. If for some $n > 0$ no rule is applicable then we set $\mathcal{I}_{n+1} = \mathcal{I}_n$. We obtain $\mathcal{I}_{C_0,\mathcal{T}}$ by setting for all concept names A and role names r:

$$\Delta^{\mathcal{I}_{C_0,\mathcal{T}}} = \bigcup_{n \geq 0} \Delta^{\mathcal{I}_n}, \quad A^{\mathcal{I}_{C_0,\mathcal{T}}} = \bigcup_{n \geq 0} A^{\mathcal{I}_n}, \quad r^{\mathcal{I}_{C_0,\mathcal{T}}} = \bigcup_{n \geq 0} r^{\mathcal{I}_n}.$$

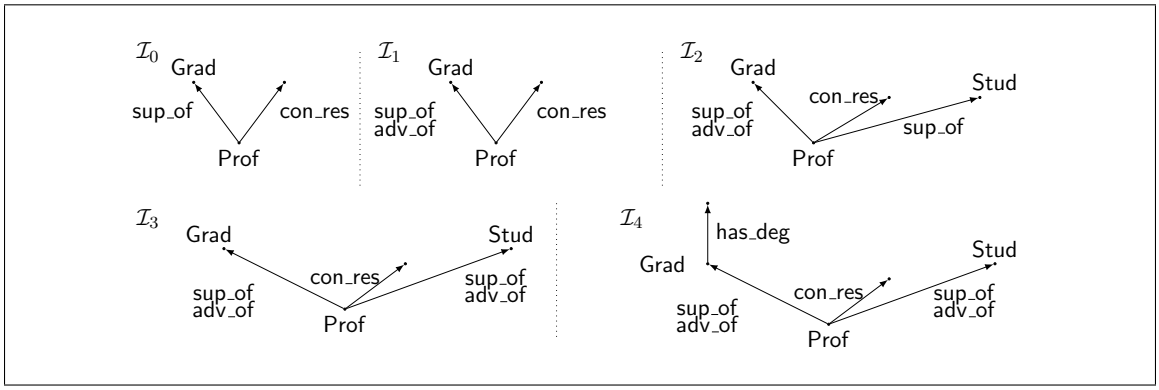

Figure 4: Canonical model construction for Example 11.

Note that the interpretation $\mathcal{I}_{C_0,\mathcal{T}}$ obtained in the limit is tree-shaped and might be infinite.[2] The following example illustrates the definition of $\mathcal{I}_{C_0,\mathcal{T}}$.

Example 11 *Consider the following TBox $\mathcal{T}$:*

$$\mathsf{Prof} \sqsubseteq \exists \mathsf{supervisor\_of}.\mathsf{Student} \tag{7}$$
$$\mathsf{Prof} \sqsubseteq \exists \mathsf{conduct\_research}.\top \tag{8}$$
$$\mathsf{Graduate} \sqsubseteq \exists \mathsf{has\_degree}.\top \tag{9}$$
$$\exists \mathsf{has\_degree}.\top \sqsubseteq \mathsf{Graduate} \tag{10}$$
$$\mathsf{supervisor\_of} \sqsubseteq \mathsf{advisor\_of} \tag{11}$$

and the concept expression

$$C_0 = \mathsf{Prof} \sqcap \exists \mathsf{conduct\_research}.\top \sqcap \exists \mathsf{supervisor\_of}.\mathsf{Graduate}.$$

Figure 4 illustrates the steps of the canonical model construction with $\mathcal{I}_0$ being $\mathcal{I}_{C_0}$ and $\mathcal{I}_4$ being the canonical model $\mathcal{I}_{C_0,\mathcal{T}}$.

The following lemma provides the announced characterisation of subsumption in the presence of TBoxes.

Lemma 12 *Let $\mathcal{T}$ be a TBox and C a concept expression. Then $\mathcal{I}_{C,\mathcal{T}}$ is a model of $\mathcal{T}$ and the following conditions are equivalent, for every concept expression D:*

1. *$\mathcal{T} \models C \sqsubseteq D$;*
2. *$\rho_C \in D^{\mathcal{I}_{C,\mathcal{T}}}$;*
3. *there is a homomorphism from T_D to $\mathcal{I}_{C,\mathcal{T}}$ that maps ρ_D to ρ_C.*

2. The exact shape of $\mathcal{I}_{C_0,\mathcal{T}}$ depends on the order of rule applications. However, all possible resulting interpretations $\mathcal{I}_{C_0,\mathcal{T}}$ are homomorphically equivalent and, as a consequence, the order of rule application is not important for our purposes.

The proof is completely standard, see, for example, the introduction by Krötzsch (2012). We only give a high-level overview. Using the construction of $\mathcal{I}_{C,\mathcal{T}}$, it is not hard to show that $\mathcal{I}_{C,\mathcal{T}}$ is a model of $\mathcal{T}$ and that $\rho_C \in C^{\mathcal{I}_{C,\mathcal{T}}}$. This implies '$1 \Rightarrow 2$' and '$2 \Rightarrow 3$' follows from Lemma 8. For '$3 \Rightarrow 1$', one can show that for any model $\mathcal{I}$ of $\mathcal{T}$ and any $d \in C^{\mathcal{I}}$, there is a homomorphism from $\mathcal{I}_{C,\mathcal{T}}$ to $\mathcal{I}$ that maps ρ_C to d. In fact, one constructs a homomorphism to $\mathcal{I}$ from each of the interpretations $\mathcal{I}_0, \mathcal{I}_1, \dots$ built during the construction of $\mathcal{I}_{C,\mathcal{T}}$, which is not hard by analysing the rules applied during that construction. The homomorphism built for each $\mathcal{I}_{n+1}$ extends that for $\mathcal{I}_n$ and thus we can take the unions of all those homomorphisms to obtain a homomorphism from $\mathcal{I}_{C,\mathcal{T}}$ to $\mathcal{I}$. It remains to compose homomorphisms and apply Lemma 8.

We are only going to use canonical models and Lemma 12 in the context of DL-Lite$^{\exists}_{\mathcal{R}}$ and DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$. Next, we identify a more subtle property of canonical models of DL-Lite$^{\exists}_{\mathcal{R}}$ and DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBoxes that we need later on. Very roughly speaking, it states a form of locality which is due to the fact that existential restrictions on the left-hand side of CIs in DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ are unqualified. Assume that $d \in (\exists r.D)^{\mathcal{I}_{C,\mathcal{T}}}$ for the canonical model $\mathcal{I}_{C,\mathcal{T}}$ of a concept expression C and TBox $\mathcal{T}$. Assume $d \in \Delta^{\mathcal{I}_C}$. We know that there exists a homomorphism h from $T_{\exists r.D}$ to $\mathcal{I}_{C,\mathcal{T}}$ mapping $\rho_{\exists r.D}$ to d. Then either h maps some elements of T_D into $\Delta^{\mathcal{I}_C}$ or it maps the whole tree T_D into $\Delta^{\mathcal{I}_{C,\mathcal{T}}} \setminus \Delta^{\mathcal{I}_C}$. We are interested in the latter case. The following lemma states that if $\mathcal{T}$ is a DL-Lite$^{\exists}_{\mathcal{R}}$ TBox, then there is a basic concept B with $d \in B^{\mathcal{I}_C}$ such that $\mathcal{T} \models B \sqsubseteq \exists r.D$. Thus, the question whether $d \in (\exists r.D)^{\mathcal{I}_{C,\mathcal{T}}}$ only depends on the concept names A with $d \in A^{\mathcal{I}_C}$ and the roles r with $d \in (\exists r.\top)^{\mathcal{I}_C}$. If $\mathcal{T}$ is a DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBox, it might not be sufficient to take a single basic concept but at least a set of basic concepts suffices (corresponding to the fact that DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ admits conjunctions on the left-hand side of CIs). This observation does not hold for $\mathcal{EL}$ TBoxes and is ultimately the reason for the fact that one cannot polynomially learn $\mathcal{EL}$ TBoxes. The following example illustrates this observation.

Example 13 *Consider the $\mathcal{EL}$ TBox $\mathcal{T} = \{\exists r.A \sqsubseteq A, A \sqsubseteq \exists r.B\}$ and let $C = \exists r.\exists r.A$. Then $\rho_C \in (\exists r.B)^{\mathcal{I}_{C,\mathcal{T}}}$ since $\mathcal{T} \models C \sqsubseteq \exists r.B$. We therefore find a homomorphism h from $T_{\exists r.B}$ to $\mathcal{I}_{C,\mathcal{T}}$ mapping $\rho_{\exists r.B}$ to ρ_C. This homomorphism maps T_B (which has a single node only) into $\Delta^{\mathcal{I}_C} \setminus \Delta^{\mathcal{I}_{C,\mathcal{T}}}$. The only basic concept B with $\rho_C \in B^{\mathcal{I}_C}$ is $\exists r.\top$ but clearly $\mathcal{T} \not\models \exists r.\top \sqsubseteq \exists r.B$ and so the observation we sketched above does not hold for $\mathcal{EL}$.*

We present this result in a more formal way. Let T_1 and T_2 be trees with labelling functions l_1 and l_2, respectively. We call T_1 a *subtree of* T_2 if the following conditions hold: $T_1 \subseteq T_2$, l_1 is the restriction of l_2 to T_1, and if $d \in T_1$ and d' is a successor of d in T_2, then d' is a successor of d in T_1 as well. The *one-neigbourhood* $N_{\mathcal{I}_C}(d)$ *of* $d \in \Delta^{\mathcal{I}_C}$ is the set of concept names A with $d \in A^{\mathcal{I}_C}$ and basic concepts $\exists r.\top$ such that there exists $d' \in \Delta^{\mathcal{I}_C}$ with $(d, d') \in r^{\mathcal{I}_C}$.

Lemma 14 *Let $\mathcal{T}$ be a DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBox, $D = \exists r.D'$ and assume $h : T_D \to \mathcal{I}_{C,\mathcal{T}}$ is such that $h(\rho_D) = d \in \Delta^{\mathcal{I}_C}$ and the image of the subtree $T_{D'}$ of T_D under h is included in $\Delta^{\mathcal{I}_{C,\mathcal{T}}} \setminus \Delta^{\mathcal{I}_C}$. Then there exists $I \subseteq N_{\mathcal{I}_C}(d)$ such that $\mathcal{T} \models \bigsqcap_{E \in I} E \sqsubseteq D$. Moreover, if $\mathcal{T}$ is a DL-Lite$^{\exists}_{\mathcal{R}}$ TBox, then there exists such a set $I \subseteq N_{\mathcal{I}_C}(d)$ with a single concept.*

Proof (sketch) This property of canonical models for DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ has been proved implicitly in many papers, for example the work of Artale et al. (2009). We give a sketch. Let N be the conjunction of all $E \in N_{\mathcal{I}_C}(d)$ and assume $\mathcal{T} \not\models N \sqsubseteq D$. Consider the canonical model $\mathcal{I}_{N,\mathcal{T}}$. By definition, the one-neighbourhoods of ρ_N in $\mathcal{I}_{N,\mathcal{T}}$ and d in $\mathcal{I}_{C,\mathcal{T}}$ coincide. Now observe that the canonical model of any concept expression C_0 and TBox $\mathcal{T}$ is obtained from $\mathcal{I}_{C_0}$ by hooking tree-shaped interpretations $\mathcal{I}_d$ with root d to every d in $\mathcal{I}_{C_0}$. As in DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ the concept expressions on the left-hand side of CIs are basic concepts, the interpretations $\mathcal{I}_d$ only depend on the one-neighbourhood $N_{\mathcal{I}_{C_0}}(d)$ of d in $\mathcal{I}_{C_0}$. Thus, the tree-shaped interpretations hooked to ρ_N in $\mathcal{I}_{N,\mathcal{T}}$ and to d in $\mathcal{I}_{C,\mathcal{T}}$ coincide and the homomorphism h given in Lemma 14 provides a homomorphism $h : T_D \to \mathcal{I}_N$ such that $h(\rho_D) = \rho_N$. By Lemma 12, $\mathcal{T} \models N \sqsubseteq D$. We have derived a contradiction. For DL-Lite$^{\exists}_{\mathcal{R}}$ one only requires a single member of $N_{\mathcal{I}_C}(d)$ since the left-hand side of CIs in DL-Lite$^{\exists}_{\mathcal{R}}$ consists of a single basic concept only. ❑

We close the introduction of description logics with some comments about the choice of our languages. In the DL literature it is not uncommon to consider the weaker variant DL-Lite$_{\mathcal{R}}$ of DL-Lite$^{\exists}_{\mathcal{R}}$ in which only basic concepts are admitted on the right-hand side of CIs, but compound concepts are not. This is often without loss of generality since every DL-Lite$^{\exists}_{\mathcal{R}}$ TBox can be expressed in DL-Lite$_{\mathcal{R}}$ by using additional role names; in this way, standard DL-Lite$^{\exists}_{\mathcal{R}}$ reasoning tasks such as subsumption and conjunctive query answering can be reduced in polynomial time to the corresponding tasks for DL-Lite$_{\mathcal{R}}$. Such a reduction is not possible in the framework of exact learning that we are concerned with in this paper. In fact, in contrast to DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes, TBoxes in DL-Lite$_{\mathcal{R}}$ are trivially polynomial time learnable using either membership queries only or equivalence queries only as there are only polynomially many CIs and RIs over a given signature.

2.3 Exact Learning

We introduce the relevant notation for exact learning. A *learning framework* $\mathfrak{F}$ is a triple $(X, \mathcal{L}, \mu)$, where X is a set of *examples* (also called *domain* or *instance space*), $\mathcal{L}$ is a set of *concepts*,[3] and μ is a mapping from $\mathcal{L}$ to 2^X. We say that $x \in X$ is a *positive example* for $l \in \mathcal{L}$ if $x \in \mu(l)$ and a *negative example* for l if $x \notin \mu(l)$.

We give a formal definition of polynomial query and time learnability within a learning framework. Let $\mathfrak{F} = (X, \mathcal{L}, \mu)$ be a learning framework. We are interested in the exact identification of a *target* concept representation $l \in \mathcal{L}$ by posing queries to oracles. Let $\mathsf{MEM}_{\mathfrak{F},l}$ be the oracle that takes as input some $x \in X$ and returns 'yes' if $x \in \mu(l)$ and 'no' otherwise. A *membership query* is a call to the oracle $\mathsf{MEM}_{\mathfrak{F},l}$. Similarly, for every $l \in \mathcal{L}$, we denote by $\mathsf{EQ}_{\mathfrak{F},l}$ the oracle that takes as input a *hypothesis* concept representation $h \in \mathcal{L}$ and returns 'yes' if $\mu(h) = \mu(l)$ and a *counterexample* $x \in \mu(h) \oplus \mu(l)$ otherwise, where $\oplus$ denotes the symmetric set difference. There is no assumption regarding which counterexample in $\mu(h) \oplus \mu(l)$ is chosen by the oracle. An *equivalence query* is a call to the oracle $\mathsf{EQ}_{\mathfrak{F},l}$.

A *learning algorithm* for $\mathfrak{F}$ is a deterministic algorithm that takes no input, is allowed to make queries to $\mathsf{MEM}_{\mathfrak{F},l}$ and $\mathsf{EQ}_{\mathfrak{F},l}$ (without knowing what the target l to be learned is), and that eventually halts and outputs some $h \in \mathcal{L}$ with $\mu(h) = \mu(l)$. We say that $\mathfrak{F}$ is

3. The similarity of this name to 'concept expression' is accidental and should not be taken to mean that these two notions are closely related. Both is standard terminology in the respective area.

exact learnable if there is a learning algorithm for $\mathfrak{F}$ and that $\mathfrak{F}$ is *polynomial query learnable* if it is exact learnable by an algorithm A such that at every step the sum of the sizes of the inputs to membership and equivalence queries made by A up to that step is bounded by a polynomial $p(|l|, |x|)$, where l is the target and $x \in X$ is the largest counterexample seen so far (Arias, 2004). Finally, $\mathfrak{F}$ is *polynomial time learnable* if it is exact learnable by an algorithm A such that at every step (we count each call to an oracle as one step of computation) of computation the time used by A up to that step is bounded by a polynomial $p(|l|, |x|)$, where $l \in \mathcal{L}$ is the target and $x \in X$ is the largest counterexample seen so far. Clearly, a learning framework $\mathfrak{F}$ that is polynomial time learnable is also polynomial query learnable.

The aim of this paper is to study learnability of description logic TBoxes. In this context, each DL L gives rise to a learning framework $(X, \mathcal{L}, \mu)$, as follows: $\mathcal{L}$ is the set of all TBoxes formulated in L, X is the set of all CIs and RIs formulated in L, and $\mu(\mathcal{T}) = \{\alpha \in X \mid \mathcal{T} \models \alpha\}$ for every $\mathcal{T} \in \mathcal{L}$. Observe that $\mu(\mathcal{T}) = \mu(\mathcal{T}')$ iff $\mathcal{T} \equiv \mathcal{T}'$, for all TBoxes $\mathcal{T}$ and $\mathcal{T}'$. We say that *L TBoxes are polynomial query learnable* if the learning framework defined by L is polynomial query learnable, and likewise for polynomial time learnability. What does not show up directly in this representation is our assumption that the signature of the target TBox is known to the learner. Note that this is a standard assumption. For example, when learning propositional Horn formulas, it is common to assume that the variables in the target formula are known to the learner.

3. Learning DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes

We prove that DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes are polynomial query learnable. If inverse roles are disallowed in CIs and RIs of the target TBox then our algorithm runs in polynomial time and thus shows that TBoxes in this restricted language are polynomial time learnable. Without this restriction, polynomial time learnability remains open.

To simplify the presentation, we make two minor assumptions about the target TBox $\mathcal{T}$. We will show later how these assumptions can be overcomed. First, we assume that $\mathcal{T}$ does not entail non-trivial role equivalences, that is, there do not exist distinct roles r and s such that $\mathcal{T} \models r \equiv s$. This allows us to avoid dealing with classes of equivalent roles, simplifying notation. The second requirement is a bit more subtle. A concept inclusion is in *reduced form* if it is between basic concepts or its left-hand side is a concept name. A TBox $\mathcal{T}$ is in *named form* if all CIs in it are in reduced form and it contains a concept name A_r such that $A_r \equiv \exists r.\top \in \mathcal{T}$, for each role r. We assume that the target TBox is in named form and that all CIs considered by the learner are in reduced form. In particular, counterexamples returned by the oracle are immediately converted into this form.

Example 15 *Although the TBox $\mathcal{T}$ from Example 11 does not entail role equivalences and all its CIs are in reduced form, it is not in named form. To fix this, we introduce concept names $A_{\mathsf{supervisor\_of}}$, $A_{\mathsf{conduct\_research}}$ and $A_{\mathsf{advisor\_of}}$ and extend $\mathcal{T}$ with the following equivalences:*

$$A_{\mathsf{supervisor\_of}} \equiv \exists \mathsf{supervisor\_of}.\top \tag{12}$$

$$A_{\mathsf{conduct\_research}} \equiv \exists \mathsf{conduct\_research}.\top \tag{13}$$

Algorithm 1 Naïve learning algorithm for DL-Lite$_{\mathcal{R}}^{\exists}$

Input: A DL-Lite$_{\mathcal{R}}^{\exists}$ TBox $\mathcal{T}$ in named form given to the oracle; $\Sigma_{\mathcal{T}}$ given to the learner.
Output: TBox $\mathcal{H}$, computed by the learner, such that $\mathcal{T} \equiv \mathcal{H}$.

1: Compute $\mathcal{H}_{basic} = \{r \sqsubseteq s \mid \mathcal{T} \models r \sqsubseteq s\} \cup \{B_1 \sqsubseteq B_2 \mid \mathcal{T} \models B_1 \sqsubseteq B_2,\ B_1, B_2 \text{ basic}\}$
2: Set $\mathcal{H}_{add} = \emptyset$
3: **while** $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\equiv \mathcal{T}$ **do**
4: Let $A \sqsubseteq C$ be the returned positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$
5: **if** there is $A \sqsubseteq C' \in \mathcal{H}_{add}$ **then**
6: Replace $A \sqsubseteq C'$ by $A \sqsubseteq C \sqcap C'$ in $\mathcal{H}_{add}$
7: **else**
8: Add $A \sqsubseteq C$ to $\mathcal{H}_{add}$
9: **end if**
10: **end while**
11: **return** $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$

$$A_{\mathsf{advisor\_of}} \quad \equiv \quad \exists \mathsf{advisor\_of}.\top. \tag{14}$$

Notice that Graduate *acts as a name for* $\exists$has_degree.$\top$ *so no new definition is needed for the role* has_degree. *The TBox* $\mathcal{T}' = \mathcal{T} \cup \{(12), (13), (14)\}$ *is in named form.*

To develop the learning algorithm it is instructive to start with a naïve version that does not always terminate but which can be refined to obtain the desired algorithm. This version is presented as Algorithm 1. Given the signature $\Sigma_{\mathcal{T}}$ of the target TBox $\mathcal{T}$, the learner starts with computing the set $\mathcal{H}_{basic}$ by posing to the oracle the membership query '$\mathcal{T} \models r \sqsubseteq s$?' for all $r, s \in \Sigma_{\mathcal{T}}$ and '$\mathcal{T} \models B_1 \sqsubseteq B_2$?' for all basic concept B_1, B_2 over $\Sigma_{\mathcal{T}}$. Observe that $\mathcal{T} \models \mathcal{H}_{basic}$. Then it enters the main **while** loop. Note that the condition '$\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\equiv \mathcal{T}$?' in Line 3 is implemented using an equivalence query to the oracle, and that $A \sqsubseteq C$ in Line 4 refers to the counterexample returned by the oracle in the case that equivalence does not hold. The counterexample must be positive since we maintain the invariant $\mathcal{T} \models \mathcal{H}_{basic} \cup \mathcal{H}_{add}$ throughout the run of the algorithm. If there is no CI of the form $A \sqsubseteq C'$ in $\mathcal{H}_{add}$ then $A \sqsubseteq C$ is added to $\mathcal{H}_{add}$, otherwise $A \sqsubseteq C \sqcap C'$ is (Lines 6 and 8). The algorithm terminates when $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \equiv \mathcal{T}$, implying that the target TBox has been learned.

Example 16 For the TBox $\mathcal{T}'$ from Example 15, Algorithm 1 first computes $\mathcal{H}_{basic}$ which coincides with $\mathcal{T}'$ except that Prof $\sqsubseteq \exists$supervisor_of.Student is not included since the concept $\exists$supervisor_of.Student is not basic. In the main loop the only counterexamples to $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \equiv \mathcal{T}'$ are (up to logical equivalence modulo $\mathcal{H}_{basic}$) the CIs

$$\mathsf{Prof} \sqsubseteq \exists \mathsf{supervisor\_of.Student}, \quad \mathsf{Prof} \sqsubseteq \exists \mathsf{advisor\_of.Student}.$$

If the oracle returns the first CI in the first iteration, the algorithm terminates immediately having learned $\mathcal{T}'$. Otherwise the oracle first returns the second CI and then returns the first CI in the second iteration. The algorithm terminates with

$$\mathcal{H}_{add} = \{\mathsf{Prof} \sqsubseteq \exists \mathsf{supervisor\_of.Student} \sqcap \exists \mathsf{advisor\_of.Student}\}$$

which is equivalent to $\mathcal{T}'$.

We now consider five examples on which this naïve algorithm fails to terminate after polynomially many steps (or at all), each example motivating a different *modification step* that is added to Algorithm 1 after Lines 4 and 5. The final, corrected algorithm is given as Algorithm 2 below. Each modification step takes as input a counterexample $A \sqsubseteq C$ against the equivalence $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \equiv \mathcal{T}$ and modifies it by posing membership queries to the oracle to obtain a CI $A' \sqsubseteq C'$ which is still a counterexample and has additional desired properties. CIs satisfying all five additional properties will be called $\mathcal{T}$*-essential*. The five modification steps are of three different types:

1. two *saturations steps*: the underlying tree of T_C is left unchanged but the labelling is modified by adding concept names to node labels or replacing roles in edge labels;

2. two *merging steps*: nodes in the tree T_C are merged, resulting in a tree with fewer nodes;

3. a *decomposition step*: T_C is replaced by a subtree or a subtree is removed from T_C, and the concept name A on the left-hand side might be replaced.

The saturation and merging steps do not change the left-hand side A of the CI $A \sqsubseteq C$ and result in a logically stronger CI $A \sqsubseteq C'$ in the sense that $\emptyset \models C' \sqsubseteq C$. In contrast, the decomposition step can be regarded as a reset operation in which also the left-hand side can change and which is logically not related to $A \sqsubseteq C$. We start with an example which motivates the first saturation step.

Example 17 *Let*

$$\mathcal{T} = \{A \sqsubseteq \exists r.A\} \cup \mathcal{T}_{\text{NF}},$$

where $\mathcal{T}_{\text{NF}} = \{A_r \equiv \exists r.\top\}$ *ensures that* $\mathcal{T}$ *is in named form. First, Algorithm 1 computes* $\mathcal{H}_{basic}$*. Afterwards the oracle can provide for the* n*-th equivalence query in the while loop the positive counterexample* $A \sqsubseteq \exists r^{n+1}.\top$*, for any* $n \geq 1$ *(here we set inductively* $\exists r^{m+1}.\top = \exists r.\exists r^m.\top$ *and* $\exists r^1.\top = \exists r.\top$*). Thus, the algorithm does not terminate.*

Informally, the problem for the learner in Example 17 is that the concepts $\exists r^n.\top$ used in the counterexamples $A \sqsubseteq \exists r^n.\top$ get larger and larger, but still none of the counterexamples implies $A \sqsubseteq \exists r.A$. We address this problem by saturating T_C with implied concept names. For the following discussion, recall that we do not distinguish between the concept expression C and its tree representation T_C. For example, if we say that C' is obtained from C by adding a concept name B to the label of node d in C, then this stands for: C' is the concept expression corresponding to the tree obtained from T_C by adding B to the label of d in T_C.

Definition 18 (Concept saturation for $\mathcal{T}$) *A CI* $A \sqsubseteq C$ *is* concept saturated for $\mathcal{T}$ *if* $\mathcal{T} \models A \sqsubseteq C$ *and* $\mathcal{T} \not\models A \sqsubseteq C'$ *for any* C' *obtained from* C *by adding a concept name to the label of some node of* C*. A CI* $A \sqsubseteq C'$ *is a* concept saturation for $\mathcal{T}$ *of a CI* $A \sqsubseteq C$ *if it is concept saturated for* $\mathcal{T}$ *and* C' *is obtained from* C *by adding concept names to the labels of some nodes in* C*.*

Observe that the learner can compute a concept saturation $A \sqsubseteq C'$ for $\mathcal{T}$ from a counterexample $A \sqsubseteq C$ by posing polynomially many membership queries to the oracle: it simply asks for any node d in T_C and concept name $E \in \Sigma_{\mathcal{T}}$ whether $\mathcal{T} \models A \sqsubseteq C^{E,d}$, where $C^{E,d}$ is obtained from C by adding E to the label of d. If the answer is positive, it replaces C by $C^{E,d}$ and proceeds. Note that there can be several concept saturations of a given CI $A \sqsubseteq C$ for a TBox $\mathcal{T}$. Consider, for example, $\mathcal{T} = \{A \sqsubseteq \exists r.(B \sqcap \exists r.\top), A \sqsubseteq \exists r.\exists r.B\}$ and the CI $A \sqsubseteq \exists r.\exists r.\top$. Then both $A \sqsubseteq A \sqcap \exists r.(B \sqcap \exists r.\top)$ and $A \sqsubseteq A \sqcap \exists r.\exists r.B$ are concept saturations of $A \sqsubseteq \exists r.\exists r.\top$ for $\mathcal{T}$ since $\mathcal{T} \not\models A \sqsubseteq \exists r.(B \sqcap \exists r.B)$.

Example 19 (Example 17 continued) *The CIs $A \sqsubseteq \exists r^n.\top$ are not concept saturated for $\mathcal{T}$. For example, for $n = 2$, a concept saturation of $A \sqsubseteq \exists r.\exists r.\top$ for $\mathcal{T}$ is given by $A \sqsubseteq A \sqcap A_r \sqcap \exists r.(A \sqcap A_r \sqcap \exists r.(A_r \sqcap A))$ (in fact, this is the only concept saturation for $\mathcal{T}$ of $A \sqsubseteq \exists r.\exists r.\top$). Now observe that if the CI $A \sqsubseteq C$ returned by the oracle to the first equivalence query is transformed by the learner into a concept saturated CI (after Line 4), then the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A\} \cup \mathcal{T}_{\text{NF}}$ is learned in one step: the only possible counterexamples returned by the oracle to the equivalence query $\mathcal{H}_{basic} \equiv \mathcal{T}$ are of the form $A \sqsubseteq C_1 \sqcap \exists r.C_2$ for some concepts C_1 and C_2. Concept saturation results in a concept of the form $C_1' \sqcap \exists r.(A \sqcap C_2')$ and $\{A \sqsubseteq C_1' \sqcap \exists r.(A \sqcap C_2')\} \models A \sqsubseteq \exists r.A$.*

The following example motivates the second saturation step. Here and in the subsequent examples we do not transform the TBoxes into named form as this does not effect the argument and simplifies presentation.

Example 20 *Consider for $n \geq 1$ the TBoxes*

$$\mathcal{T}_n \quad = \quad \{A \sqsubseteq \exists e_1.\exists e_2. \ldots \exists e_n.\top\} \cup \{e_i \sqsubseteq r_i, e_i \sqsubseteq s_i \mid 1 \leq i \leq n\}.$$

For $M \subseteq \{1, \ldots, n\}$, set $C_M = \exists t_1.\exists t_2. \ldots \exists t_n.\top$, where $t_i = r_i$ if $i \in M$ and $t_i = s_i$ if $i \notin M$. Then for the first 2^n equivalence queries in the while loop the oracle can provide a positive counterexample $A \sqsubseteq C_M$ by always choosing a fresh set $M \subseteq \{1, \ldots, n\}$.

Intuitively, the problem for the learner in Example 20 is that there are exponentially many logically incomparable CIs that are entailed by $\mathcal{T}_n$ but do not entail $A \sqsubseteq \exists e_1.\exists e_2. \ldots \exists e_n.\top$. A step towards resolving this problem is to replace the roles r_i and s_i by the roles e_i in the counterexamples $A \sqsubseteq C_M$.

Definition 21 (Role saturation for $\mathcal{T}$) *A CI $A \sqsubseteq C$ is* role saturated for $\mathcal{T}$ *if $\mathcal{T} \models A \sqsubseteq C$ and $\mathcal{T} \not\models A \sqsubseteq C'$ for any C' obtained from C by replacing in some edge label a role r by a role $s \neq r$ with $\mathcal{T} \models s \sqsubseteq r$. A CI $A \sqsubseteq C'$ is a* role saturation for $\mathcal{T}$ *of a CI $A \sqsubseteq C$ if it is role saturated for $\mathcal{T}$ and C' is obtained from C by replacing in some edge labels a role r by a role s with $\mathcal{T} \models s \sqsubseteq r$.*

Similarly to concept saturation, the learner can compute a role saturated $A \sqsubseteq C'$ from a counterexample $A \sqsubseteq C$ by posing polynomially many membership queries. Again, there can be several role saturations of a given $A \sqsubseteq C$ for a TBox $\mathcal{T}$. For example, if

$$\mathcal{T} = \{s_1 \sqsubseteq r, s_2 \sqsubseteq r, A \sqsubseteq \exists s_1.B, A \sqsubseteq \exists s_2.B\},$$

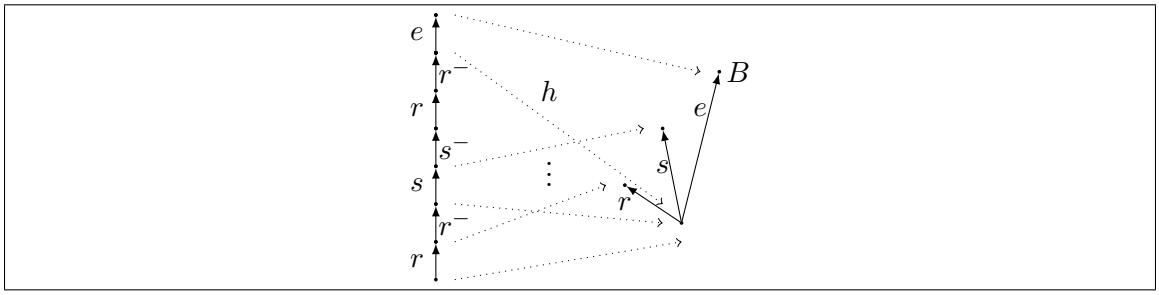

Figure 5: Tree representation of $C_{\{1,3\}}$ and homomorphism to $\exists r.\top \sqcap \exists s.\top \sqcap \exists e.B$.

then $A \sqsubseteq \exists s_1.B$ and $A \sqsubseteq \exists s_2.B$ are role saturations of $A \sqsubseteq \exists r.B$ for $\mathcal{T}$. Observe that in Example 20 the single role saturation of any $A \sqsubseteq C_M$ is $A \sqsubseteq \exists e_1.\exists e_2.\ldots.\exists e_n.\top$. Thus, if the counterexample $A \sqsubseteq C_M$ returned by the first equivalence query is transformed into a role saturated CI, then the algorithm terminates after one step. We now introduce and motivate our two merging rules.

Example 22 Consider the TBox

$$\mathcal{T} = \{A \sqsubseteq \exists r.\top \sqcap \exists s.\top \sqcap \exists e.B\}.$$

and fix an $n \geq 1$. For $M \subseteq \{1, \ldots, n\}$, set $C_M = \exists t_1.\exists t_1^-.\exists t_2.\exists t_2^-.\ldots.\exists t_n.\exists t_n^-.\exists e.\top$, where $t_i = r$ if $i \in M$ and $t_i = s$ if $i \notin M$. Figure 5 (left) illustrates the concept expression $C_{\{1,3\}}$, assuming $n = 3$. By Lemma 10, $\mathcal{T} \models A \sqsubseteq C_M$ since there is a homomorphism h from T_{C_M} to the labelled tree that corresponds to $\exists r.\top \sqcap \exists s.\top \sqcap \exists e.B$, as shown in Figure 5. Thus, the oracle can provide for the first 2^n equivalence queries a positive counterexample $A \sqsubseteq C_M$ by always choosing a fresh set $M \subseteq \{1, \ldots, n\}$.

The problem for the learner in Example 22 is similar to that in Example 20: there are exponentially many logically incomparable CIs that are entailed by $\mathcal{T}$ but do not entail $A \sqsubseteq \exists r.\top \sqcap \exists s.\top \sqcap \exists e.B$. A step towards solving this problem is to merge the predecessor and successor nodes of a node if the edge labels are inverse to each other and the resulting CI is still implied by the TBox.

Definition 23 (Parent/Child Merging for $\mathcal{T}$) *A concept C' is* obtained from a concept C by parent/child merging *if C' is obtained from C by choosing nodes d, d', d'' such that d is an r-successor of d', and d'' is an r^--successor of d, for some role r, and then removing d'', setting $l(d') = l(d') \cup l(d'')$, and making every s-successor e of d'' in C an s-successor of d', for any role s.*

Let $A \sqsubseteq C$ be a CI with $\mathcal{T} \models A \sqsubseteq C$. A CI $A \sqsubseteq C'$ is obtained from $A \sqsubseteq C$ by parent/child merging *if $\mathcal{T} \models A \sqsubseteq C'$ and C' is obtained from C by parent/child merging. We say that $A \sqsubseteq C$ is* parent/child merged for $\mathcal{T}$ *if there is no $A \sqsubseteq C'$ with $C \neq C'$ that can be obtained from $A \sqsubseteq C$ by parent/child merging.*

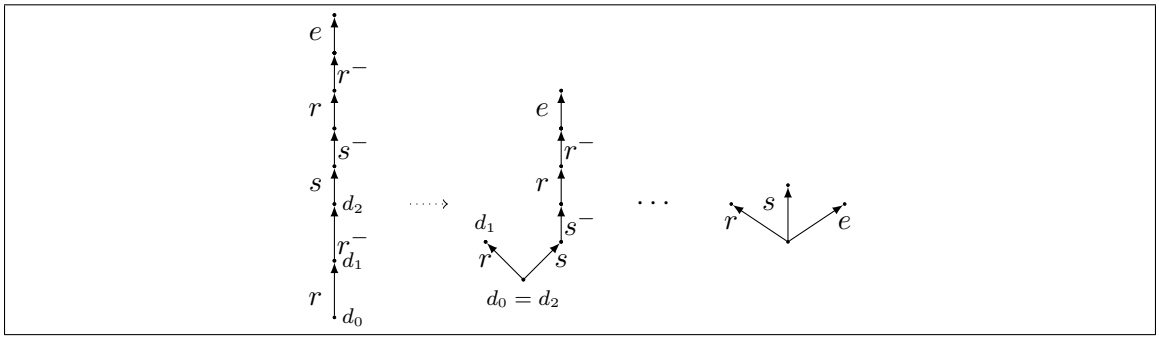

Figure 6: Parent/Child Merging of $C_{\{1,3\}}$ for $\mathcal{T}$

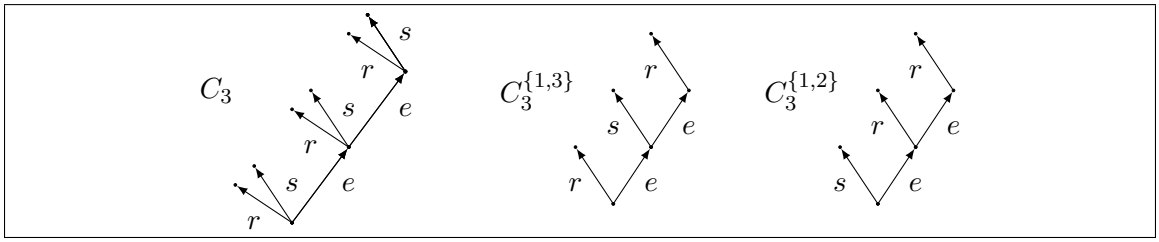

Figure 7: Tree representation of the concept expressions C_3, $C_3^{\{1,3\}}$ and $C_3^{\{1,2\}}$.

Note that when C' is obtained from C by parent/child merging with d' and d'' as in Definition 23, then $\emptyset \models C' \sqsubseteq C$. To show this, one can use Lemma 10 and the natural homomorphism h from T_C to $T_{C'}$, that is, the identity except that $h(d'') = d'$.

Similarly to the saturation operations, the learner can compute a parent/child merged $A \sqsubseteq C'$ by posing polynomially many membership queries. In Example 22 the parent/child merging of any $A \sqsubseteq C_M$ with $\emptyset \neq M \neq \{1, \dots, n\}$ is $A \sqsubseteq \exists r.\top \sqcap \exists s.\top \sqcap \exists e.\top$, as illustrated in Figure 6: in the first step the nodes d_0 and d_2 are merged, two additional merging steps give $\exists r.\top \sqcap \exists s.\top \sqcap \exists e.\top$. The following example motivates the second merging operation.

Example 24 *Define concept expressions C_i by induction as follows:*

$$C_1 = \exists r.\top \sqcap \exists s.\top, \quad C_{i+1} = C_1 \sqcap \exists e.C_i$$

and let

$$\mathcal{T}_n = \{A \sqsubseteq \exists e.C_n\}.$$

For $M \subseteq \{1, \dots, n\}$, set $C_1^M = \exists r.\top$ if $1 \in M$ and $C_1^M = \exists s.\top$ if $1 \notin M$. Also, let $C_{i+1}^M = \exists r.\top \sqcap \exists e.C_i^M$ if $i+1 \in M$ and $C_{i+1}^M = \exists s.\top \sqcap \exists e.C_i^M$ if $i+1 \notin M$, $1 \leq i < n$. Figure 7 illustrates concept expressions of the form C_n and C_n^M.

As an answer to the first 2^n equivalence queries the oracle can compute a positive counterexample $A \sqsubseteq \exists e.C_n^M$ by always choosing a fresh set $M \subseteq \{1, \dots, n\}$.

To deal with this example we introduce a modification step that identifies siblings in C rather than a parent and a child.

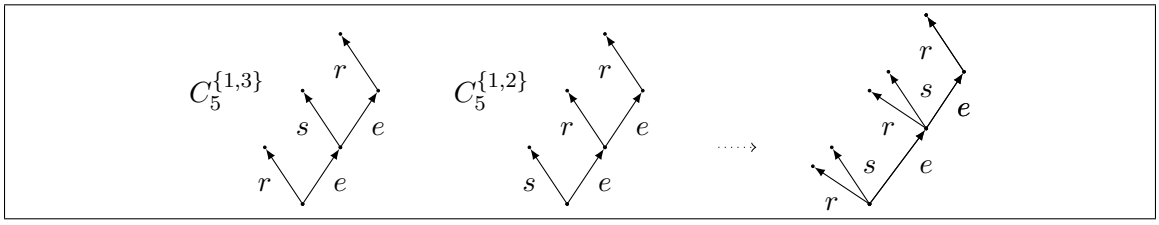

Figure 8: Sibling Merging $C_3^{\{1,3\}}$ and $C_3^{\{1,2\}}$ for $\mathcal{T}$.

Definition 25 (Sibling Merging for $\mathcal{T}$) *A concept C' is* obtained from a concept C by sibling merging *if C' is obtained from C by choosing nodes d, d', d'' such that d' and d'' are r-successors of d, for some role r, and then removing d'', setting $l(d') = l(d') \cup l(d'')$, and making every s-successor e of d'' in T_C an s-successor of d', for any role s.*

Let $A \sqsubseteq C$ be a CI with $\mathcal{T} \models A \sqsubseteq C$. A CI $A \sqsubseteq C'$ is obtained from $A \sqsubseteq C$ by sibling merging *if $\mathcal{T} \models A \sqsubseteq C'$ and C' is obtained from C by sibling merging. We say that $A \sqsubseteq C$ is* sibling merged for $\mathcal{T}$ *if there is no $A \sqsubseteq C'$ with $C \neq C'$ that can be obtained from $A \sqsubseteq C$ by sibling merging.*

It can be verified that when C' is obtained from C by sibling merging, then $\emptyset \models C' \sqsubseteq C$.

In Example 24 the counterexamples $A \sqsubseteq C_n^M$ are actually sibling merged for $\mathcal{T}_n$. Thus, producing a sibling merged $A \sqsubseteq C'$ directly from the counterexamples returned by the oracle does not overcome the problem illustrated by the example. Instead, we apply sibling merging after Line 5 of the algorithm: instead of adding $A \sqsubseteq C \sqcap C'$ to $\mathcal{H}_{add}$, the learner computes a sibling merged $A \sqsubseteq D$ from this CI and adds it to $\mathcal{H}_{add}$. For Example 24, this is illustrated in Figure 8. Clearly, after at most $n + 1$ counterexamples, the learner has added $A \sqsubseteq \exists e.C_n$, as required.

Finally, we need a decomposition rule. The following variant of Example 17 illustrates that the four modification steps introduced so far do not yet lead to a polynomial learning algorithm even if they are applied both after Line 4 and after Line 5 in Algorithm 1.

Example 26 *Let*

$$\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq \exists r.B\}.$$

The oracle can provide for the n-th equivalence query the positive counterexample $A \sqsubseteq C_{B,n}$, where $C_{B,n} = A \sqcap D_{B,n}$ and, inductively, $D_{B,0} = B$ and $D_{B,n+1} = B \sqcap \exists r.D_{B,n}$, for any $n \geq 0$. The algorithm does not terminate even with the four modification steps introduced above applied after Lines 4 and 5: the CIs $A \sqsubseteq C_{B,n}$ are concept and role saturated and they are parent/child and sibling merged.

The problem illustrated in Example 26 is that so far the learning algorithm attempts to learn $\mathcal{T}$ without ever considering to add to $\mathcal{H}_{add}$ a CI whose left-hand side is B (rather than A). To deal with this problem we introduce a 'reset step' that, in contrast to the previous modification steps, can lead to a different left-hand side and also to a CI that does not imply the original CI given $\mathcal{T}$, as in all previous modification steps.

Definition 27 (Decomposed CI for $\mathcal{T}$) *Let $A \sqsubseteq C$ be a CI with $\mathcal{T} \models A \sqsubseteq C$. We say that $A \sqsubseteq C$ is* decomposed for $\mathcal{T}$ *if for every non-root node d in C, every concept name*

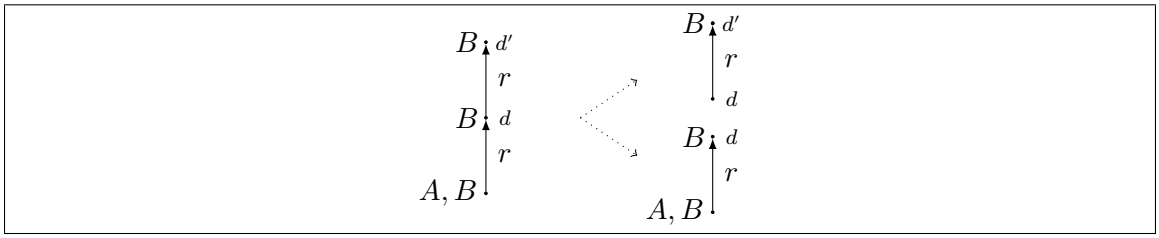

Figure 9: Illustration of decomposition of CIs. Here $C|^{-}_{d'\downarrow} = A \sqcap B \sqcap \exists r.B$ and $C' = B$.

$A' \in l(d)$, and every r-successor d' of d in C, we have $\mathcal{T} \not\models A' \sqsubseteq \exists r.C'$ where C' corresponds to the subtree of C rooted at d'.

In contrast to the previous four modification steps, the membership queries used by the learner to obtain a decomposed CI do not only depend on $\mathcal{T}$ but also on the hypothesis $\mathcal{H}_{add} \cup \mathcal{H}_{basic}$ computed up to that point: starting from CI $A \sqsubseteq C$, the learner takes a non-root node d in C, a concept name $A' \in l(d)$, and an r-successor d' of d in C, and then checks using a membership query whether $\mathcal{T} \models A' \sqsubseteq \exists r.C'$, where C' is the subtree rooted at d' in C. If the check succeeds, $A \sqsubseteq C$ is replaced by

(a) $A' \sqsubseteq \exists r.C'$ if $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A' \sqsubseteq \exists r.C'$; and otherwise by

(b) $A \sqsubseteq C|^{-}_{d'\downarrow}$, where $C|^{-}_{d'\downarrow}$ is obtained from C by removing the subtree rooted in d' from C.

Note that $\{A \sqsubseteq C|^{-}_{d'\downarrow}, A' \sqsubseteq \exists r.C'\} \models A \sqsubseteq C$. Thus, one of the CIs $A \sqsubseteq C|^{-}_{d'\downarrow}$ and $A' \sqsubseteq \exists r.C'$ is not entailed by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$, and this is the CI that replaces the original CI.

In Example 26, assume that the oracle returns $A \sqsubseteq C$ with $C = A \sqcap B \sqcap \exists r.(B \sqcap \exists r.B)$ as the first counterexample. The tree T_C corresponding to C is shown on the left-hand side of Figure 9. This CI is not decomposed for $\mathcal{T}$: the label of node d contains B, the concept C' rooted in d' in C is B and $\mathcal{T} \models B \sqsubseteq \exists r.B$. Since $\mathcal{H} \cup \mathcal{H}_{add} \not\models B \sqsubseteq \exists r.B$, Case (a) applies and $A \sqsubseteq B$ is replaced by $B \sqsubseteq \exists r.B$.

This finishes the description of the modification steps. It turns out that they cure all problems with the initial version of the algorithm and enable polynomial query learnability.

Definition 28 *A DL-Lite$^{\exists}_{\mathcal{R}}$ CI is* $\mathcal{T}$-essential *if it is concept saturated, role saturated, parent/child merged, sibling merged, and decomposed for $\mathcal{T}$.*

After Lines 4 and 5 of Algorithm 1, we need to make the CI currently considered $\mathcal{T}$-essential, by exhaustively applying the modification steps described above in all possible orders. The resulting refined version of the learning algorithm is shown as Algorithm 2. We next analyse the properties of this algorithm.

Polynomial Query Bound on the Algorithm

If Algorithm 2 terminates, then it obviously has found a TBox $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ that is logically equivalent to $\mathcal{T}$. It thus remains to show that the algorithm terminates after polynomially many polynomial size queries. Observe that $\mathcal{H}_{add}$ contains at most one CI $A \sqsubseteq C$ for each concept name A. At each step in the while loop, either some $A' \sqsubseteq C'$ is added to $\mathcal{H}_{add}$ such

Algorithm 2 The learning algorithm for DL-Lite$^{\exists}_{\mathcal{R}}$

Input: A DL-Lite$^{\exists}_{\mathcal{R}}$ TBox $\mathcal{T}$ in named form given to the oracle; $\Sigma_{\mathcal{T}}$ given to the learner.
Output: TBox $\mathcal{H}$, computed by the learner, such that $\mathcal{T} \equiv \mathcal{H}$.

1: Compute $\mathcal{H}_{basic} = \{r \sqsubseteq s \mid \mathcal{T} \models r \sqsubseteq s\} \cup \{B_1 \sqsubseteq B_2 \mid \mathcal{T} \models B_1 \sqsubseteq B_2,\ B_1, B_2 \text{ basic}\}$
2: Set $\mathcal{H}_{add} = \emptyset$
3: **while** $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\equiv \mathcal{T}$ **do**
4: Let $A \sqsubseteq C$ be the returned positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$
5: Find a $\mathcal{T}$-essential CI $A' \sqsubseteq C'$ such that $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A' \sqsubseteq C'$
6: **if** there is $A' \sqsubseteq C'' \in \mathcal{H}_{add}$ **then**
7: Find $\mathcal{T}$-essential CI $A' \sqsubseteq C^*$ such that $\emptyset \models C^* \sqsubseteq C'' \sqcap C'$
8: Replace $A' \sqsubseteq C''$ by $A' \sqsubseteq C^*$ in $\mathcal{H}_{add}$
9: **else**
10: Add $A' \sqsubseteq C'$ to $\mathcal{H}_{add}$
11: **end if**
12: **end while**
13: **return** $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$

that no CI with A' on the left-hand side existed in $\mathcal{H}_{add}$ before (Line 10) or an existing CI $A' \sqsubseteq C''$ in $\mathcal{H}_{add}$ is replaced by a fresh CI $A' \sqsubseteq C^*$ with $\emptyset \models C^* \sqsubseteq C''$.

We start with showing that Lines 5 and 7 can be implemented with polynomially many membership queries. The next lemma addresses Line 5.

Lemma 29 *Given a positive counterexample $A \sqsubseteq C$ for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$, one can construct a $\mathcal{T}$-essential counterexample $A' \sqsubseteq C'$ using only polynomially many polynomial size membership queries in $|C| + |\mathcal{T}|$.*

Proof Let $A \sqsubseteq C$ be a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ and assume the five modification steps introduced above are applied exhaustively by posing membership queries to the oracle. Observe that the number of applications of modifications steps is bounded polynomially in $|C| \times |\mathcal{T}|$. To show this, let n_C be the number of nodes in T_C. Then $n_{C''} = n_{C'}$ if $A'' \sqsubseteq C''$ is obtained from $A' \sqsubseteq C'$ by a concept or role saturation step and $n_{C''} < n_{C'}$ if $A'' \sqsubseteq C''$ is obtained from $A' \sqsubseteq C'$ by a merging or decomposition step. Thus, the number of applications of merging and decomposition steps is bounded by n_C and the number of applications of concept and role saturated steps is bounded by $|\Sigma_{\mathcal{T}}| \times n_C$ and $|\Sigma_{\mathcal{T}}| \times n_C^2$, respectively. Thus, after at most $n_C + |\Sigma_{\mathcal{T}}| \times n_C + |\Sigma_{\mathcal{T}}| \times n_C^2$ steps no modification step is applicable and the final CI is $\mathcal{T}$-essential. We verify that it is also a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. It suffices to show that the CI resulting from each single modification step is entailed by $\mathcal{T}$, but not by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. The former has been shown when we introduced the modification steps. Regarding the latter, in the first four modification steps we have $\mathcal{H}_{basic} \models C' \sqsubseteq C$ if $A \sqsubseteq C$ is replaced by $A \sqsubseteq C'$. Hence $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A \sqsubseteq C'$. For the decomposition step, we have already argued, after Definition 27, that the added CI is not entailed by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. ❏

The following lemma addresses Line 7.

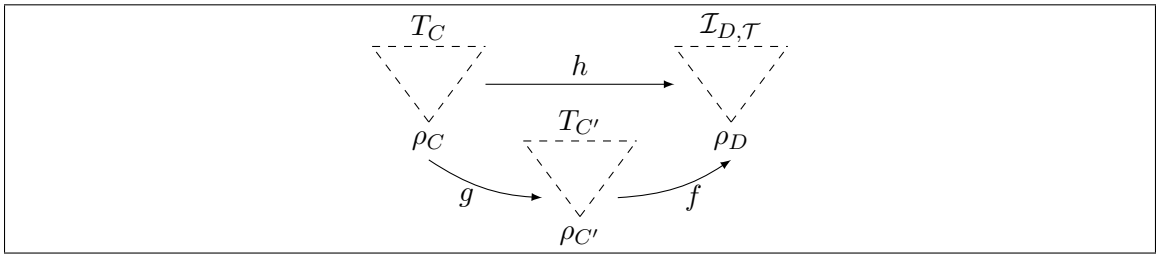

Figure 10: Homomorphisms in Lemma 31.

Lemma 30 *Assume that $A \sqsubseteq C_1$ and $A \sqsubseteq C_2$ are $\mathcal{T}$-essential. Then one can construct a $\mathcal{T}$-essential $A \sqsubseteq C$ such that $\emptyset \models C \sqsubseteq C_1 \sqcap C_2$ using polynomially many polynomial size membership queries in $|C_1| + |C_2|$.*

Proof We start with $A \sqsubseteq C_1 \sqcap C_2$. Using the fact that C_1 and C_2 are both $\mathcal{T}$-essential, one can show that this CI is (i) concept saturated for $\mathcal{T}$, (ii) role saturated for $\mathcal{T}$, (iii) parent/child merged for $\mathcal{T}$, and (iv) decomposed for $\mathcal{T}$. Assume, for example, that $A \sqsubseteq C_1 \sqcap C_2$ is not concept saturated. Then one can add a new concept name A' to the label $l(d)$ for some node d in $T_{C_1 \sqcap C_2}$ and $\mathcal{T} \models A \sqsubseteq C'$ for the resulting concept C'. Clearly d is a node in T_{C_1} or in T_{C_2}. Assume without loss of generality that d is in T_{C_1} and let C_1' be the concept obtained from C_1 by adding A' to $l(d)$. Then $\mathcal{T} \models A \sqsubseteq C_1'$ since $\mathcal{T} \models A \sqsubseteq C'$ which contradicts the assumption that $A \sqsubseteq C_1$ is concept saturated. The remaining three modification steps are considered similarly. We now exhaustively apply the modification step 'Sibling merging for $\mathcal{T}$' and use the resulting CI as the desired $A \sqsubseteq C$. Similarly to the argument above one can show that a CI with properties (i)–(iv) still has those properties after applying sibling merging. Thus, $A \sqsubseteq C$ is $\mathcal{T}$-essential. We have argued in the proof of Lemma 29 already that the number of applications of a sibling merging step to a CI of the form $A \sqsubseteq D$ is bounded by the number of nodes in T_D. Thus, the number of modification steps is bounded polynomially in $|C_1| + |C_2|$. ❑

To analyse the algorithm further we first prove a polynomial upper bound on the size of $\mathcal{T}$-essential CIs. To this end, we require the notion of an isomorphic embedding and an auxiliary lemma. A homomorphism $h : T_C \to \mathcal{I}$ is an *isomorphic embedding for $\mathcal{T}$* if it is injective, $A \in l(d)$ if $h(d) \in A^{\mathcal{I}}$ for all concept names A, and for $r = l(d, d')$ it holds that $\mathcal{T} \models r \sqsubseteq s$ for all $(h(d), h(d')) \in s^{\mathcal{I}}$. The following lemma shows that for $\mathcal{T}$-essential CIs $A \sqsubseteq C$ and any D that interpolates between A and C (meaning that $\mathcal{T} \models A \sqsubseteq D$ and $\mathcal{T} \models D \sqsubseteq C$) the homomorphism h from T_C to $\mathcal{I}_{D,\mathcal{T}}$ that witnesses $\rho_D \in C^{\mathcal{I}_{D,\mathcal{T}}}$ (see Lemma 12) is an isomorphic embedding.

Lemma 31 *Assume the $A \sqsubseteq C$ is $\mathcal{T}$-essential, $\mathcal{T} \models A \sqsubseteq D$ and $\mathcal{T} \models D \sqsubseteq C$. Then any homomorphism $h : T_C \to \mathcal{I}_{D,\mathcal{T}}$ that maps ρ_C to ρ_D is an isomorphic embedding.*

Proof Assume first that h is not injective. Then there is a parent/child or sibling merging C' of C and a homomorphism $f : T_{C'} \to \mathcal{I}_{D,\mathcal{T}}$ such that $h = f \circ g$ for the natural homomorphism $g : T_C \to T_{C'}$ (Figure 10). By Lemma 12, $\mathcal{T} \models D \sqsubseteq C'$. Thus, $\mathcal{T} \models A \sqsubseteq C'$ and we have derived a contradiction to the assumption that C is parent/child and sibling merged.

Now let T' be the following labelled tree: the nodes in T' are the same as in T, $A \in l'(d)$ iff $h(d) \in A^{\mathcal{I}_{D,\mathcal{T}}}$, and for any two nodes d, d' with d' a successor of d: $l'(d, d') = r$ for the unique role r with $\mathcal{T} \models r \sqsubseteq s$ for all s with $(d, d') \in s^{\mathcal{I}_{D,\mathcal{T}}}$. Let C' be the concept expression that corresponds to T'. Then $\rho_D \in C'^{\mathcal{I}_{D,\mathcal{T}}}$ and so, by Lemma 12, $\mathcal{T} \models D \sqsubseteq C'$. Thus $\mathcal{T} \models A \sqsubseteq C'$. But then $A \sqsubseteq C'$ can be obtained from $A \sqsubseteq C$ by concept and role saturation steps. As $A \sqsubseteq C$ is concept and role saturated already, $C = C'$ and so h is an isomorphic embedding. ❏

We are now able to prove that $\mathcal{T}$-essential CIs are of polynomial size in $\mathcal{T}$. For a concept C, let n_C denote the number of nodes in the tree representation T_C of C and let

$$A^{\mathcal{T}} = \{A\} \cup \{B \mid A \sqsubseteq B \in \mathcal{H}_{basic}\} \cup \{D \mid \mathcal{T} \models A \sqsubseteq B, B \sqsubseteq D \in \mathcal{T}\}.$$

Lemma 32 *If $A \sqsubseteq C$ is $\mathcal{T}$-essential, then $n_C \leq \sum_{D \in A^{\mathcal{T}}} n_D$.*

Proof Assume $A \sqsubseteq C$ is $\mathcal{T}$-essential. Let $D_0 := \bigsqcap_{D \in A^{\mathcal{T}}} D$ and $\mathcal{I}_{D_0,\mathcal{T}}$ be the canonical model of D_0 and $\mathcal{T}$. By Lemma 12, there is a homomorphism $h : T_C \to \mathcal{I}_{D_0,\mathcal{T}}$ mapping ρ_C to ρ_{D_0}. By Lemma 31, h is an isomorphic embedding. Using that $A \sqsubseteq C$ is decomposed for $\mathcal{T}$, we now show that h maps T_C into the restriction of $\mathcal{I}_{D_0,\mathcal{T}}$ to $\Delta^{\mathcal{I}_{D_0}}$ from which the lemma follows since h is injective.

For a proof by contradiction, assume that there exists d' in T_C with $h(d') \notin \Delta^{\mathcal{I}_{D_0}}$. As $h(\rho_C) \in \Delta^{\mathcal{I}_{D_0}}$, we may assume that all d' on the path from ρ_C to d' are mapped to $\Delta^{\mathcal{I}_{D_0}}$ (if this is not the case, we can replace d' by the first element on the path from ρ_C to d' not mapped into $\Delta^{\mathcal{I}_{D_0}}$). In particular, the parent d of d' in T_C is mapped into $\Delta^{\mathcal{I}_{D_0}}$. Let $l(d, d') = r$. Observe that the whole subtree rooted in d' must be mapped into $\Delta^{\mathcal{I}_{D_0,\mathcal{T}}} \setminus \Delta^{\mathcal{I}_{D_0}}$ since otherwise h would not be injective.

Let $C' = \exists r.C''$, where C'' corresponds to the subtree rooted in d' in C. By Lemma 14, there exists a basic concept B such that $h(d) \in B^{\mathcal{I}_{D_0,\mathcal{T}}}$ and $\mathcal{T} \models B \sqsubseteq C'$. As $\mathcal{T}$ is in named form there exists a concept name E with $\mathcal{T} \models E \equiv B$. Thus, $h(d) \in E^{\mathcal{I}_{D_0,\mathcal{T}}}$ and $\mathcal{T} \models E \sqsubseteq C'$. As h is an isomorphic embedding, $E \in l(d)$. We make a case distinction:

- $h(d) \neq \rho_{D_0}$. Then $A \sqsubseteq C$ is not decomposed for $\mathcal{T}$ since C contains an edge (d, d') such that E is in the node label of d, $l(d, d') = r$, and $\mathcal{T} \models E \sqsubseteq \exists r.C''$. We have derived a contradiction.

- $h(d) = \rho_{D_0}$. As $h(d') \in \Delta^{\mathcal{I}_{D_0,\mathcal{T}}} \setminus \Delta^{\mathcal{I}_{D_0}}$, by construction of $\mathcal{I}_{D_0,\mathcal{T}}$, there exists a CI $B_0 \sqsubseteq D \in \mathcal{T}$ with B_0 a basic concept such that $\rho_{D_0} \in B_0^{\mathcal{I}_{D_0,\mathcal{T}}}$ and $h(d')$ is in the copy of the tree-shaped interpretation $\mathcal{I}_D$ which was attached to ρ_{D_0} in the construction of $\mathcal{I}_{D_0,\mathcal{T}}$. But since $\mathcal{T} \models A \sqsubseteq B_0$ we have $D \in A^{\mathcal{T}}$ and so $\rho_{D_0} \in D^{\mathcal{I}_{D_0}}$. But then, by the construction of $\mathcal{I}_{D_0,\mathcal{T}}$, no fresh $\mathcal{I}_D$ was attached to ρ_{D_0} because D is already satisfied in $\mathcal{I}_{D_0}$ and we have derived a contradiction.

❏

We are now in the position to prove that the learning algorithm terminates after posing a polynomial number of queries.

Lemma 33 *For every concept name A, the number of replacements of a CI $A \sqsubseteq C$ in $\mathcal{H}_{add}$ by a CI of the form $A \sqsubseteq C'$ is bounded polynomially in $|\mathcal{T}|$.*

Proof We aim to show that when $A \sqsubseteq C$ is replaced with $A \sqsubseteq C'$, then the number of nodes in the tree representation of C' is strictly larger than the number of nodes in the tree representation of C. Since any CI $A \sqsubseteq C$ ever added to $\mathcal{H}_{add}$ is $\mathcal{T}$-essential, Lemma 32 then yields that the number of replacements is bounded by $\sum_{D \in A^{\mathcal{T}}} n_D$, which is polynomial in $|\mathcal{T}|$.

A straightforward analysis of Algorithm 2 reveals that when $A \sqsubseteq C$ is replaced with $A \sqsubseteq C'$, then $\emptyset \models C' \sqsubseteq C$ and $\emptyset \not\models C \sqsubseteq C'$ (otherwise the positive counterexample returned by the oracle would be a consequence of $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$). Moreover, both $A \sqsubseteq C$ and $A \sqsubseteq C'$ are consequences of $\mathcal{T}$. It thus suffices to establish the following.

Claim. If $A \sqsubseteq C$ is $\mathcal{T}$-essential, $\mathcal{T} \models A \sqsubseteq C'$, $\emptyset \models C' \sqsubseteq C$, and $\emptyset \not\models C \sqsubseteq C'$, then T_C is obtained from $T_{C'}$ by removing at least one subtree.

We prove the claim. Since $\emptyset \models C' \sqsubseteq C$, by Lemma 12 there is a homomorphism h from T_C to the canonical model $\mathcal{I}_{C'}$ that maps ρ_C to $\rho_{C'}$. Then h is also a homomorphism into the canonical model $\mathcal{I}_{C',\mathcal{T}}$. Since $A \sqsubseteq C$ is $\mathcal{T}$-essential, $\mathcal{T} \models A \sqsubseteq C'$, and $\emptyset \models C' \sqsubseteq C$, Lemma 31 yields that h is an isomorphic embedding into $\mathcal{I}_{C',\mathcal{T}}$. Then, trivially, h is also an isomorphic embedding into $\mathcal{I}_{C'}$ which means that T_C is obtained from $T_{C'}$ by removing subtrees. Since $\emptyset \not\models C \sqsubseteq C'$, at least one subtree must in fact have been removed. ❑

We have obtained the following main result of this section.

Theorem 34 *DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes are polynomial query learnable using membership and equivalence queries. Moreover, DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes without inverse roles can be learned in polynomial time using membership and equivalence queries.*

Proof Recall that our algorithm requires the target TBox to be in named form. We first show Theorem 34 under that assumption and then argue that the assumption can be dropped.

In each iteration of Algorithm 2, either a CI is added to $\mathcal{H}_{add}$ or a CI is replaced in $\mathcal{H}_{add}$. Since the number of times the former happens is bounded by $|\Sigma_{\mathcal{T}}|$ and (by Lemma 33) the number of times the latter happens is polynomial in $|\mathcal{T}|$, the number of iterations of Algorithm 2 is polynomial in $|\mathcal{T}|$. For polynomial query learnability of DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes, it remains to show that in each iteration Algorithm 2 makes only polynomially many polynomial size queries in $|\mathcal{T}|$ and the size of the largest counterexample seen so far. We start with equivalence queries, made only in Line 3. We have already argued that the number of iterations is polynomial in $|\mathcal{T}|$ and thus so is the number of equivalence queries made. Regarding their size, we observe that there are at most $|\Sigma_{\mathcal{T}}|^2$ CIs in $\mathcal{H}_{basic}$ and at most $|\Sigma_{\mathcal{T}}|$ CIs in $\mathcal{H}_{add}$, that the size of CIs in $\mathcal{H}_{basic}$ is constant and by Lemma 32 the size of CIs in $\mathcal{H}_{add}$ is polynomial in $|\mathcal{T}|$. Membership queries are made only in Lines 5 and 7 for which it suffices to invoke Lemmas 29 and 30.

Now for the "moreover" part of Theorem 34. Observe that since each (membership or equivalence) query counts as one step of computation, the only potentially costly step of Algorithm 2 is the implementation of the decomposition step in Line 5, which relies on making subsumption checks of the form $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \models A \sqsubseteq C$. As discussed in Section 2, deciding subsumption in DL-Lite$_{\mathcal{R}}^{\exists}$ is NP-complete while in $\mathcal{EL}$ with role inclusions subsumption is in PTime. As DL-Lite$_{\mathcal{R}}^{\exists}$ without inverse roles is a fragment of $\mathcal{EL}$ with role inclusions, we obtain polynomial time learnability for TBoxes in this case.

To drop the requirement that the target TBox is in named form, we show that any polynomial (query or time) learning algorithm for TBoxes in named form can be transformed into the same kind of algorithm for unrestricted target TBoxes. In fact, the learner can use at most $\mathcal{O}(|\Sigma_{\mathcal{T}}|^2)$ membership queries "Does $\mathcal{T}$ entail $r \sqsubseteq s$?" to compute for every role r the class $[r]_{\mathcal{T}}$ of roles s with $\mathcal{T} \models s \equiv r$ and choose a representative $r_{\mathcal{T}}$ for this class. Then whenever some $s \in [r]_{\mathcal{T}}$ is used in any counterexample returned by the oracle, it gets replaced with $r_{\mathcal{T}}$. Likewise, whenever $\mathcal{T}$ does not have a name for some $\exists r.\top$, the algorithm still uses the concept name A_r in its internal representations (although they are no longer included in the signature $\Sigma_{\mathcal{T}}$ of the target TBox) and replaces $\exists r.\top$ with A_r in the counterexamples returned by the oracle. It also replaces each A_r with $\exists r.\top$ in membership queries to the oracle and in the hypothesis used for posing equivalence queries. ❑

4. Learning DL-Lite$_{\mathcal{R},\mathsf{horn}}^{\exists}$ TBoxes

We study exact learnability of TBoxes in DL-Lite$_{\mathcal{R},\mathsf{horn}}^{\exists}$, the extension of DL-Lite$_{\mathcal{R}}^{\exists}$ that admits conjunctions of basic concepts on the left-hand side of CIs. This language is a generalisation of both DL-Lite$_{\mathcal{R}}^{\exists}$ and propositional Horn logic. In fact, the algorithm we present combines the classical algorithms for propositional Horn logic (Angluin et al., 1992; Frazier and Pitt, 1993) with the algorithm for DL-Lite$_{\mathcal{R}}^{\exists}$ presented in Section 3. The resulting algorithm is quite subtle and indeed this is the reason why we treated the DL-Lite$_{\mathcal{R}}^{\exists}$ case separately in Section 3.

To simplify the presentation, we make the same assumptions as in Section 3 about the target TBox $\mathcal{T}$ with signature $\Sigma_{\mathcal{T}}$. In particular, we assume that $\mathcal{T}$ is in named form, suitably generalised to DL-Lite$_{\mathcal{R},\mathsf{horn}}^{\exists}$: there are no distinct roles r and s such that $\mathcal{T} \models r \equiv s$, for each role r the TBox $\mathcal{T}$ contains an equivalence $A_r \equiv \exists r.\top$ and all CIs of $\mathcal{T}$ are either CIs between basic concepts or contain no concept expressions of the form $\exists r.\top$ on the left-hand side, for any role r. Denote by $\mathsf{lhs}(\alpha)$ the set of concept names that occur as conjuncts on the left-hand side of a CI α and denote by $\mathsf{rhs}(\alpha)$ the set of concept expressions that occur as top-level conjuncts on the right-hand side of α (that is, they are not nested inside restrictions). We often do not distinguish between the set $\mathsf{lhs}(\alpha)$ and the conjunction over all its concept expressions, and similarly for $\mathsf{rhs}(\alpha)$. For example, if $\alpha_1 = C_1 \sqsubseteq D_1$ and $\alpha_2 = C_2 \sqsubseteq D_2$ then $\mathsf{lhs}(\alpha_1) \sqsubseteq \mathsf{rhs}(\alpha_2)$ stands for $C_1 \sqsubseteq D_2$. Also, if $\mathsf{lhs}(\alpha_1) = \{A_1, A_2, A_3\}$ and $\mathsf{lhs}(\alpha_2) = \{A_2, A_3, A_4\}$ then $\mathsf{lhs}(\alpha_1) \cap \mathsf{lhs}(\alpha_2) \sqsubseteq D$ stands for $A_2 \sqcap A_3 \sqsubseteq D$.

The algorithm for learning DL-Lite$_{\mathcal{R},\mathsf{horn}}^{\exists}$ TBoxes is shown as Algorithm 3. Like Algorithm 2, Algorithm 3 first determines the set $\mathcal{H}_{basic}$ that contains all CIs $B_1 \sqsubseteq B_2$ with B_1, B_2 basic concepts such that $\mathcal{T} \models B_1 \sqsubseteq B_2$ and all RIs $r \sqsubseteq s$ such that $\mathcal{T} \models r \sqsubseteq s$. The hypothesis $\mathcal{H}$ is the union of $\mathcal{H}_{basic}$ and $\mathcal{H}_{add}$. In contrast to Algorithm 2, $\mathcal{H}_{add}$ is an *ordered list* of CIs rather than a set. We write α_i to denote the CI α at position i in the list $\mathcal{H}_{add}$. In

Algorithm 3 The learning algorithm for DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBoxes

Input: A DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBox $\mathcal{T}$ in named form given to the oracle; $\Sigma_{\mathcal{T}}$ given to the learner.
Output: TBox $\mathcal{H}$, computed by the learner, such that $\mathcal{T} \equiv \mathcal{H}$.

1: Compute $\mathcal{H}_{basic} = \{r \sqsubseteq s \mid \mathcal{T} \models r \sqsubseteq s\} \cup \{B_1 \sqsubseteq B_2 \mid \mathcal{T} \models B_1 \sqsubseteq B_2,\ B_1, B_2 \text{ basic}\}$
2: Set $\mathcal{H}_{add}$ to be the empty list and $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$
3: **while** $\mathcal{H} \not\equiv \mathcal{T}$ **do**
4: Let γ be the returned positive counterexample for $\mathcal{T}$ and $\mathcal{H}$
5: Find a $\mathcal{T}$-essential γ' with $\mathcal{H} \not\models \gamma'$ and $|\{\exists r.F \mid \exists r.F \in \mathsf{rhs}(\gamma')\}| \leq 1$
6: Left saturate γ' for $\mathcal{H}$
7: **if** there is $A \in \mathsf{N_C}$ such that $\mathcal{T} \models \mathsf{lhs}(\gamma') \sqsubseteq A$ and $\mathcal{H} \not\models \mathsf{lhs}(\gamma') \sqsubseteq A$ **then**
8: $\mathcal{H}$:=CN-Refine$(\mathcal{H}, \mathsf{lhs}(\gamma') \sqsubseteq A)$
9: **else**
10: $\mathcal{H}$:=$\exists$-Refine$(\mathcal{H}, \gamma')$
11: **end if**
12: Set $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$
13: **end while**
14: **return** $\mathcal{H}$

the learning algorithm, working with an ordered list of CIs allows the learner to pick the first α_i in $\mathcal{H}_{add}$ with a certain property and merge it with a new CI, a technique we adopt from the work of Angluin et al. (1992) and Frazier and Pitt (1993). As in Algorithm 2, $\mathcal{T} \models \mathcal{H}$ is a loop invariant, thus, γ is necessarily positive. The algorithm terminates when $\mathcal{H} \equiv \mathcal{T}$.

Algorithm 4 Function CN-Refine$(\mathcal{H}, \gamma)$

1: **if** there is $A \in \mathsf{N_C}$ and $\alpha_i \in \mathcal{H}_{add}$ such that $\mathcal{T} \models \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq A$,
2: and $\mathcal{H} \not\models \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq A$ **then**
3: Concept saturate $\gamma' = \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq A$ for $\mathcal{T}$
4: Replace the first such α_i in $\mathcal{H}_{add}$ by γ'
5: **else**
6: Concept saturate γ for $\mathcal{T}$
7: Append γ to the list $\mathcal{H}_{add}$
8: **end if**
9: **return** $\mathcal{H}$

Algorithm 3 uses membership queries to compute a $\mathcal{T}$-essential counterexample γ such that $\mathsf{rhs}(\gamma)$ contains at most one concept expression of the form $\exists r.F$ (Line 5) and which is 'Left saturated for $\mathcal{H}$' (Line 6); here, a CI is left saturated for $\mathcal{H}$ if its left-hand side contains all subsuming concept names w.r.t. $\mathcal{H}$ (Definition 35 below) and $\mathcal{T}$-essential if it satisfies the conditions for $\mathcal{T}$-essential CIs from Section 3, appropriately modified for CIs with conjunctions of concept names on the left-hand side (Definition 37 below). Then, the algorithm checks whether there is a concept name A such that $\mathsf{lhs}(\gamma) \sqsubseteq A$ is a positive counterexample. If so, then it calls Function CN-Refine (Algorithm 4) and updates the hypothesis either by refining some α_i in $\mathcal{H}_{add}$ or by appending a new CI to $\mathcal{H}_{add}$. The

Algorithm 5 Function $\exists$-Refine$(\mathcal{H}, \gamma)$

1: **if** there is $C \in \mathsf{rhs}(\gamma)$ of the form $\exists r.D$ and $\alpha_i \in \mathcal{H}_{add}$ such that $\mathcal{T} \models \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq C$
2: and $\mathcal{H} \not\models \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq C$ **then**
3: **if** $\mathcal{T} \models \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq C \sqcap \mathsf{rhs}(\alpha_i)$ **then**
4: Find a $\mathcal{T}$-essential $\mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq D^*$ with $\emptyset \models D^* \sqsubseteq C \sqcap \mathsf{rhs}(\alpha_i)$
5: Replace the first such α_i in $\mathcal{H}_{add}$ by $\mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq D^*$
6: **else**
7: Concept saturate $\gamma' = \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq C$ for $\mathcal{T}$
8: Replace the first such α_i in $\mathcal{H}_{add}$ by γ'
9: **end if**
10: **else**
11: Append γ to the list $\mathcal{H}_{add}$
12: **end if**
13: **return** $\mathcal{H}$

number of replacements of any given α_i in $\mathcal{H}_{add}$ in CN-Refine is bounded by $|\Sigma_\mathcal{T}|$ since whenever α_i is replaced in CN-Refine$(\mathcal{H}, \gamma)$, then $\mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \subsetneq \mathsf{lhs}(\alpha_i)$.[4]

If there is no concept name A such that $\mathsf{lhs}(\gamma) \sqsubseteq A$ is a positive counterexample then Algorithm 3 calls Function $\exists$-Refine (Algorithm 5). In this case one considers the existential restrictions that occur on top-level on the right-hand side of γ. Note that $\exists$-Refine can be viewed as a variation of the body of the while loop in Algorithm 2 in which one considers sets of concept names on the left-hand side of CIs rather than a single concept name. Recall that in Algorithm 2, the new CI γ and a CI α in $\mathcal{H}_{add}$ are merged if they have the same concept name on the left-hand side. In contrast, now they are merged if the intersection of their left-sides is still subsumed by some existential restriction C from $\mathsf{rhs}(\gamma)$ (Lines 1 and 2). There are two cases: if the intersection is also subsumed by $\mathsf{rhs}(\alpha)$ (checked in Line 3), then in the next line a $\mathcal{T}$-essential counterexample is computed and the first such α_i is replaced by the new CI. Otherwise it follows that $\mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \subsetneq \mathsf{lhs}(\alpha_i)$ and the first such α_i is replaced by the CI computed in Line 7. Note that the latter can happen at most $|\Sigma_\mathcal{T}|$ times for each CI in $\mathcal{H}_{add}$ (and the former can happen at most $|\mathcal{T}|$ times for each CI in $\mathcal{H}_{add}$, see Lemma 43 below). If no CI can be refined with γ then $\exists$-Refine appends γ to $\mathcal{H}_{add}$.

We now define the step 'left-saturate γ for $\mathcal{H}$' used in Line 6 of Algorithm 3. Observe that this step is meaningless for DL-Lite$_\mathcal{R}^\exists$.

Definition 35 (Left saturation for $\mathcal{H}$) *A CI γ' is obtained from a CI γ by* left saturation for $\mathcal{H}$ *if* $\mathsf{rhs}(\gamma') = \mathsf{rhs}(\gamma)$ *and* $\mathsf{lhs}(\gamma') = \{A \in \Sigma_\mathcal{T} \mid \mathcal{H} \models \mathsf{lhs}(\gamma) \sqsubseteq A\}$. *A CI γ is* left saturated for $\mathcal{H}$ *if it coincides with its left saturation for $\mathcal{H}$.*

One can clearly left saturate any CI γ for $\mathcal{H}$ by checking whether $\mathcal{H} \models \mathsf{lhs}(\gamma) \models A$ for every $A \in \Sigma_\mathcal{T}$. The following example shows that Line 6 is necessary for Algorithm 3 to be polynomial. A similar step is also necessary in Frazier et al.'s algorithm learning propositional Horn logic from entailments (Frazier and Pitt, 1993).

4. This is a consequence of the fact that $\mathcal{H}_{add}$ only contains concept saturated CIs (defined essentially as in the previous section, see Definition 37 below): $\mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) = \mathsf{lhs}(\alpha_i)$ and $\mathcal{T} \models \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq A$ implies $A \in \mathsf{rhs}(\alpha_i)$ by concept saturatedness, thus contradicting $\mathcal{H} \not\models \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq A$.

Example 36 Assume Line 6 of Algorithm 3 is omitted. Let for $n \geq 2$,

$$\mathcal{T}_n \quad = \quad \{E_1 \sqcap \cdots \sqcap E_n \sqsubseteq A\} \cup \{A_i \sqsubseteq E_i, B_i \sqsubseteq E_i \mid 1 \leq i \leq n\}.$$

For $M \subseteq \{1, \ldots, n\}$, set $C_M = \bigsqcap_{i \leq n} C_i$, where $C_i = A_i$ if $i \in M$ and $C_i = B_i$ if $i \notin M$. Then the oracle can provide for the first 2^n equivalence queries in the while loop of Algorithm 3 a positive counterexample $C_M \sqsubseteq A$ by always choosing a fresh set $M \subseteq \{1, \ldots, n\}$.

For the refinements on the right-hand side, we extend to DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ the notion of $\mathcal{T}$-essential CIs introduced in the previous section:

1. (Concept saturation for $\mathcal{T}$) A CI γ is *concept saturated for* $\mathcal{T}$ if $\mathcal{T} \models \gamma$ and $\mathcal{T} \not\models \gamma'$ for any γ' with $\mathsf{lhs}(\gamma) = \mathsf{lhs}(\gamma')$ such that $\mathsf{rhs}(\gamma')$ is obtained from $\mathsf{rhs}(\gamma)$ by adding a concept name to the label of some node of $\mathsf{rhs}(\gamma)$. A CI γ' is a *concept saturation for* $\mathcal{T}$ of a CI γ if it is concept saturated for $\mathcal{T}$, $\mathsf{lhs}(\gamma) = \mathsf{lhs}(\gamma')$, and $\mathsf{rhs}(\gamma')$ is obtained from $\mathsf{rhs}(\gamma)$ by adding concept names to the labels of some nodes of $\mathsf{rhs}(\gamma)$.

2. (Role saturation for $\mathcal{T}$) A CI γ is *role saturated for* $\mathcal{T}$ if $\mathcal{T} \models \gamma$ and $\mathcal{T} \not\models \gamma'$ for any γ' with $\mathsf{lhs}(\gamma) = \mathsf{lhs}(\gamma')$ such that $\mathsf{rhs}(\gamma')$ is obtained from $\mathsf{rhs}(\gamma)$ by replacing in some edge label a role r by a role $s \neq r$ with $\mathcal{T} \models s \sqsubseteq r$. A CI γ' is a *role saturation for* $\mathcal{T}$ of a CI γ if it is role saturated for $\mathcal{T}$, $\mathsf{lhs}(\gamma) = \mathsf{lhs}(\gamma')$, and $\mathsf{rhs}(\gamma')$ is obtained from $\mathsf{rhs}(\gamma)$ by replacing in some edge labels a role r by a role $s \neq r$ with $\mathcal{T} \models s \sqsubseteq r$.

3. (Parent/child merged for $\mathcal{T}$) A CI γ' is *obtained from a CI* γ *by parent/child merging for* $\mathcal{T}$ if $\mathsf{lhs}(\gamma) = \mathsf{lhs}(\gamma')$, $\mathsf{rhs}(\gamma')$ is obtained from $\mathsf{rhs}(\gamma')$ by parent/child merging (as in Definition 23), and $\mathcal{T} \models \gamma'$. A CI γ is *parent/child merged for* $\mathcal{T}$ if $\mathcal{T} \models \gamma$ and there is no γ' with $\gamma \neq \gamma'$ that can be obtained from γ by parent/child merging for $\mathcal{T}$.

4. (Sibling merged for $\mathcal{T}$) A CI γ' is *obtained from a CI* γ *by sibling merging for* $\mathcal{T}$ if $\mathsf{lhs}(\gamma) = \mathsf{lhs}(\gamma')$, $\mathsf{rhs}(\gamma')$ is obtained from $\mathsf{rhs}(\gamma')$ by sibling merging (as in Definition 25), and $\mathcal{T} \models \gamma'$. A CI γ is *sibling merged for* $\mathcal{T}$ if $\mathcal{T} \models \gamma$ and there is no γ' with $\gamma \neq \gamma'$ that can be obtained from γ by sibling merging for $\mathcal{T}$.

5. (Decomposed CI for $\mathcal{T}$) A CI γ is *decomposed for* $\mathcal{T}$ if $\mathcal{T} \models \gamma$ and for every non-root node d in $\mathsf{rhs}(\gamma)$, every role r, and every r-successor d' of d in $\mathsf{rhs}(\gamma)$ we have $\mathcal{T} \not\models l(d) \sqsubseteq \exists r.C'$, where C' corresponds to the subtree of $\mathsf{rhs}(\gamma)$ rooted at d'.

Definition 37 *A DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ CI is* $\mathcal{T}$-essential *if it is concept saturated, role saturated, parent/child merged, sibling merged, and decomposed for* $\mathcal{T}$*.*

The saturation, merging and decomposition steps defined above are straightforward generalisations of Definitions 18 to 27 to CIs with conjunctions on the left-hand side. One can easily generalise the arguments from DL-Lite$^{\exists}_{\mathcal{R}}$ to show that for any CI γ with $\mathcal{H} \not\models \gamma$ one can compute a $\mathcal{T}$-essential γ' with $\mathcal{H} \not\models \gamma'$ using polynomially many membership queries and entailment checks relative to $\mathcal{H}$. For the analysis of the learning algorithm it is crucial that all CIs in the ordered list $\mathcal{H}_{add}$ are $\mathcal{T}$-essential at all times, which we prove next.

Lemma 38 *At any point in the execution of Algorithm 3, all CIs in* $\mathcal{H}_{add}$ *are* $\mathcal{T}$*-essential.*

Proof If a CI γ is of the form $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A$ with A a concept name, then the set $\mathsf{rhs}(\gamma')$ of the concept saturation γ' of γ for $\mathcal{T}$ contains concept names only. Thus, γ' is $\mathcal{T}$-essential. It follows that the CIs added to $\mathcal{H}_{add}$ in Lines 4 and 7 of CN-Refine are $\mathcal{T}$-essential. Also, it is easy to see that if γ is $\mathcal{T}$-essential and $C \in \mathsf{rhs}(\gamma)$ then the concept saturation of $\mathsf{lhs}(\gamma) \sqsubseteq C$ for $\mathcal{T}$ is $\mathcal{T}$-essential as well. Thus, the CI γ' in Line 8 of $\exists$-Refine is $\mathcal{T}$-essential. ❏

Polynomial Query Bound on the Algorithm

As in the previous section it is immediate that upon termination the algorithm has found a TBox $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$ that is logically equivalent to the target TBox $\mathcal{T}$. It thus remains to show that it issues only polynomially many queries of polynomial size. We first discuss how Lines 5 and 6 of Algorithm 3 and Line 4 of $\exists$-Refine can be implemented. The next lemma addresses Lines 5 and 6 of Algorithm 3.

Lemma 39 *Given a positive counterexample γ for $\mathcal{T}$ relative to $\mathcal{H}$, one can construct with polynomially many polynomial size membership queries in $|\gamma|$ and $|\mathcal{T}|$, a counterexample γ' that is left saturated for $\mathcal{H}$, $\mathcal{T}$-essential and such that $|\{\exists r.F \mid \exists r.F \in \mathsf{rhs}(\gamma')\}| \leq 1$.*

The proof of Lemma 39 is a straightforward extension of the proof of Lemma 29 and uses the observation that a left-saturated γ' for $\mathcal{H}$ can be computed from γ by adding all concept names $A \in \Sigma_{\mathcal{T}}$ with $\mathcal{H} \models \mathsf{lhs}(\gamma) \sqsubseteq A$ to $\mathsf{lhs}(\gamma)$. This lemma also requires that $|\{\exists r.F \mid \exists r.F \in \mathsf{rhs}(\gamma')\}| \leq 1$. If there is $A \in \mathsf{rhs}(\gamma')$ such that $\mathcal{H} \not\models \mathsf{lhs}(\gamma') \sqsubseteq A$ then we can simply drop all conjuncts of the form $\exists r.F$ from $\mathsf{rhs}(\gamma')$. Otherwise, we can satisfy the condition by simply choosing a conjunct $\exists r.F \in \mathsf{rhs}(\gamma')$ such that $\mathcal{H} \not\models \mathsf{lhs}(\gamma') \sqsubseteq \exists r.F$ and then apply 'Concept saturation for $\mathcal{T}$' to $\mathsf{lhs}(\gamma') \sqsubseteq \exists r.F$. The resulting γ' is left saturated for $\mathcal{H}$, $\mathcal{T}$-essential and has at most one conjunct of the form $\exists r.F$ in $\mathsf{rhs}(\gamma')$.

The following lemma addresses Line 4 of $\exists$-Refine.

Lemma 40 *Assume that α and γ are $\mathcal{T}$-essential and there is $C \in \mathsf{rhs}(\gamma)$ such that $\mathcal{T} \models \mathsf{lhs}(\alpha) \cap \mathsf{lhs}(\gamma) \sqsubseteq \mathsf{rhs}(\alpha) \sqcap C$. Then one can construct, with polynomially many polynomial size membership queries in $|\mathsf{rhs}(\alpha)|$ and $|C|$, a $\mathcal{T}$-essential $\mathsf{lhs}(\alpha) \cap \mathsf{lhs}(\gamma) \sqsubseteq D^*$ such that $\emptyset \models D^* \sqsubseteq \mathsf{rhs}(\alpha) \sqcap C$.*

Proof Assume $\mathcal{T} \models \mathsf{lhs}(\alpha) \cap \mathsf{lhs}(\gamma) \sqsubseteq \mathsf{rhs}(\alpha) \sqcap C$. Then, similar to Lemma 30, one can show that the only property of $\mathcal{T}$-essential CIs that can fail is being sibling merged for $\mathcal{T}$ and that after applying the step 'Sibling merging for $\mathcal{T}$' to $\mathsf{lhs}(\alpha) \cap \mathsf{lhs}(\gamma) \sqsubseteq \mathsf{rhs}(\alpha) \sqcap C$ the resulting CI is $\mathcal{T}$-essential, as required. ❏

We also have to show that the number of CIs in $\mathcal{H}_{add}$ is bounded polynomially in $|\mathcal{T}|$ and for each position of $\mathcal{H}_{add}$ the number of replacements is bounded polynomially in $|\mathcal{T}|$. These properties follow from the following lemma.

Lemma 41 *Let $\mathcal{H}_{add}$ be a ordered list of CIs computed at some point of an execution of Algorithm 3. Then*

(i) the length of $\mathcal{H}_{add}$ is bounded by the number of CIs in $\mathcal{T}$ and

(ii) The number of replacements of an existing CI $\alpha \in \mathcal{H}_{add}$ is bounded polynomially in $|\mathcal{T}|$.

The rest of the section is devoted to proving Lemma 41. We first show Point (ii) of Lemma 41 and start by generalising Lemma 32 on the size of $\mathcal{T}$-essentials CIs. For any conjunction C of concept names we set

$$\begin{aligned} C^{\mathcal{T}} &= \{D \mid \mathcal{T} \models C \sqsubseteq A_1 \sqcap \cdots \sqcap A_k \text{ and } A_1 \sqcap \cdots \sqcap A_k \sqsubseteq D \in \mathcal{T}\} \cup \\ &\quad \{B \mid \mathcal{T} \models C \sqsubseteq B, B \text{ basic concept over } \Sigma_{\mathcal{T}}\} \end{aligned}$$

Recall that for any concept expression C we denote by n_C the number of nodes in the tree T_C corresponding to C.

Lemma 42 *If α is $\mathcal{T}$-essential, then $n_{\mathsf{rhs}(\alpha)} \leq \sum_{D \in \mathsf{lhs}(\alpha)^{\mathcal{T}}} n_D$.*

Proof The proof is almost the same as the proof of Lemma 32. Assume α is $\mathcal{T}$-essential. Let $D_0 := \bigsqcap_{D \in \mathsf{lhs}(\alpha)^{\mathcal{T}}} D$ and let $\mathcal{I}_{D_0,\mathcal{T}}$ be the canonical model of D_0 and $\mathcal{T}$. Now one can prove in almost the same way as in the proof of Lemma 32 that the homomorphism h from $T_{\mathsf{rhs}(\alpha)}$ into $\mathcal{I}_{D_0,\mathcal{T}}$ mapping $\rho_{\mathsf{rhs}(\alpha)}$ to $\rho_{D_0,\mathcal{T}}$ is an injective mapping into $\mathcal{I}_{D_0}$ (using Lemma 14 for DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ instead of DL-Lite$^{\exists}_{\mathcal{R}}$). ❑

We are now in the position to prove Point (ii) of Lemma 41.

Lemma 43 *The number of replacements of an existing CI $\alpha \in \mathcal{H}_{add}$ is bounded polynomially in $|\mathcal{T}|$.*

Proof A CI $\alpha \in \mathcal{H}_{add}$ can be replaced in Line 4 of CN-Refine or in Lines 5 or 8 of $\exists$-Refine. If α is replaced by α' in Line 4 of CN-Refine or in Line 8 of $\exists$-Refine then $\mathsf{lhs}(\alpha') \subsetneq \mathsf{lhs}(\alpha)$, so the number of replacements is bounded by $|\Sigma_{\mathcal{T}}|$. If α is replaced by α' in Line 5 of $\exists$-Refine, then either $\mathsf{lhs}(\alpha') \subsetneq \mathsf{lhs}(\alpha)$ or $\mathsf{lhs}(\alpha') = \mathsf{lhs}(\alpha)$. For the latter case one can show as in the proof of Lemma 33 for DL-Lite$^{\exists}_{\mathcal{R}}$, the following

Claim. If $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq C$ and $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq C'$ are $\mathcal{T}$-essential, and $\emptyset \models C' \sqsubseteq C$, then T_C is obtained from $T_{C'}$ by removing subtrees.

Thus, each time $\alpha \in \mathcal{H}_{add}$ is replaced in Line 5 of $\exists$-Refine without decreasing the number of concept names in $\mathsf{lhs}(\alpha)$, the number $n_{\mathsf{rhs}(\alpha)}$ of nodes in the tree representation of $\mathsf{rhs}(\alpha)$ strictly increases. By Lemma 42, $n_{\mathsf{rhs}(\alpha)}$ is bounded polynomially in $|\mathcal{T}|$ and the lemma follows. ❑

We now come to the proof of Point (i) of Lemma 41. To formulate an upper bound on the length of $\mathcal{H}_{add}$ in terms of $\mathcal{T}$ it is convenient to assume that the right-hand side of every CI in $\mathcal{T}$ is *primitive*, that is, either a concept name or a concept expression of the form $\exists r.D$. This assumption is w.l.o.g. since one can equivalently transform every CI $C \sqsubseteq D_1 \sqcap D_2$ into two CIs $C \sqsubseteq D_1$ and $C \sqsubseteq D_2$. We call such a TBox *rhs-primitive*. Note that CIs in $\mathcal{H}$ may still have multiple concepts on the right-hand side.

A concept C is called *concept saturated for* $\mathcal{T}$ if $\mathcal{T} \models C \sqsubseteq C'$ whenever C' results from C by adding a new concept name A' to the label of some node in T_C. Denote by C^{sat} the (unique) concept obtained from C by adding concept names to the node labels of T_C until it is concept saturated for $\mathcal{T}$. The following definition enables us to link the CIs in $\mathcal{H}_{add}$ to the CIs in $\mathcal{T}$.

Definition 44 *Let $\mathcal{T}$ be rhs-primitive. We say that a CI α* has target $\beta \in \mathcal{T}$ *if*

1. $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha)$ *and*
2. *there exists $D \in \mathsf{rhs}(\alpha) \setminus \mathsf{lhs}(\alpha)$ such that $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq D$.*

We aim to show that Algorithm 3 maintains the invariant that

(iii) every $\alpha \in \mathcal{H}_{add}$ has some target $\beta \in \mathcal{T}$ and

(iv) every $\beta \in \mathcal{T}$ is the target of at most one $\alpha \in \mathcal{H}_{add}$.

Then Point (i) of Lemma 41 clearly follows.

Example 45 To illustrate Definition 44, suppose that

$$\mathcal{T} = \{A_1 \sqcap A_4 \sqsubseteq A_2,\ A_2 \sqsubseteq \exists r.A_3,\ A_3 \sqsubseteq A_4,\ A_r \equiv \exists r.\top\}$$

is the target TBox. $\mathcal{T}$ is rhs-primitive. To simplify notation, we use β_i to denote the i-th CI occurring in $\mathcal{T}$ above. Assume $\mathcal{H}_{basic} = \{\beta_3, \beta_4\}$ and $\mathcal{H}_{add} = \emptyset$. Let $\alpha_1 = A_1 \sqcap A_3 \sqsubseteq A_2$. Then there is no $\beta_i \in \mathcal{T}$ such that α_1 has target β_i. However, by applying left saturation for $\mathcal{H}$ to α_1 we obtain $\alpha_1' = A_1 \sqcap A_3 \sqcap A_4 \sqsubseteq A_2$ and since $\mathsf{lhs}(\beta_1) \subseteq \mathsf{lhs}(\alpha_1')$ and $A_2 \notin \mathsf{lhs}(\alpha_1')$, α_1' has target β_1. For $\alpha_2 = A_1 \sqcap A_4 \sqsubseteq \exists r.A_3$, there is no $\beta_i \in \mathcal{T}$ such that α_2 has target β_i. But α_2 is not $\mathcal{T}$-essential and making it $\mathcal{T}$-essential results in $\alpha_2' = A_1 \sqcap A_4 \sqsubseteq A_r \sqcap A_1 \sqcap A_2 \sqcap A_4 \sqcap \exists r.(A_3 \sqcap A_4)$ which again has target β_1. Finally, let $\alpha_3 = A_2 \sqsubseteq \exists r.A_4$. As $\mathsf{lhs}(\beta_2) \subseteq \mathsf{lhs}(\alpha_3)$ and $\emptyset \models A_r \sqcap \exists r.(A_3 \sqcap A_4) \sqsubseteq \exists r.A_4$, α_3 has target β_2. Note that α_3 is not $\mathcal{T}$-essential, but the result of making it $\mathcal{T}$-essential also has target β_2.

Point (iii) is a consequence of the following lemma.

Lemma 46 *Let $\mathcal{T}$ be rhs-primitive and let γ be a $\mathcal{T}$-essential CI such that $\emptyset \not\models \gamma$. Then γ has some target $\beta \in \mathcal{T}$.*

Proof Assume γ is $\mathcal{T}$-essential and $\emptyset \not\models \gamma$. Assume for a proof by contradiction that γ has no target in $\mathcal{T}$. We first show the following

Claim 1. If γ has no target in $\mathcal{T}$ and $\mathcal{T} \models \mathsf{lhs}(\gamma) \sqsubseteq A$ then $A \in \mathsf{lhs}(\gamma)$, for all $A \in \mathsf{N_C}$.

For the proof of Claim 1, consider the canonical model $\mathcal{I}_{\mathsf{lhs}(\gamma),\mathcal{T}}$ of $\mathsf{lhs}(\gamma)$ and $\mathcal{T}$. Recall that $\rho_{\mathsf{lhs}(\gamma),\mathcal{T}}$ denotes the root of $\mathcal{I}_{\mathsf{lhs}(\gamma),\mathcal{T}}$. By Lemma 12, $\rho_{\mathsf{lhs}(\gamma),\mathcal{T}} \in D^{\mathcal{I}_{\mathsf{lhs}(\gamma),\mathcal{T}}}$ iff $\mathcal{T} \models \mathsf{lhs}(\gamma) \sqsubseteq D$, for any concept D. Thus, it suffices to prove that $\rho_{\mathsf{lhs}(\gamma),\mathcal{T}} \in A^{\mathcal{I}_{\mathsf{lhs}(\gamma),\mathcal{T}}}$ implies $A \in \mathsf{lhs}(\gamma)$, for all concept names A. The proof is by induction over the sequence $\mathcal{I}_0, \ldots$ used to construct $\mathcal{I}_{\mathsf{lhs}(\gamma),\mathcal{T}}$, where $\mathcal{I}_0 = \mathcal{I}_{\mathsf{lhs}(\gamma)}$. For $\mathcal{I}_{\mathsf{lhs}(\gamma)}$ this is the case by definition. Now suppose the claim holds for $\mathcal{I}_n$ and $\rho_{\mathsf{lhs}(\gamma),\mathcal{T}} \in A^{\mathcal{I}_{n+1}} \setminus A^{\mathcal{I}_n}$. Then there either exist concept names $A_1, \ldots, A_k$ with $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq A \in \mathcal{T}$ and $\rho_{\mathsf{lhs}(\gamma),\mathcal{T}} \in (A_1 \sqcap \cdots \sqcap A_k)^{\mathcal{I}_n}$ or there exists $\exists r.\top$ with $\exists r.\top \sqsubseteq A \in \mathcal{T}$ and $\rho_{\mathsf{lhs}(\gamma),\mathcal{T}} \in (\exists r.\top)^{\mathcal{I}_n}$. In the first case, we have $\{A_1, \ldots, A_k\} \subseteq \mathsf{lhs}(\gamma)$ by induction hypothesis and so $A \in \mathsf{lhs}(\gamma)$ because otherwise $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq A$ would be a target of γ. In the second case there must be an $\mathcal{I}_m$ with $m < n$ such that there are $E_1 \sqcap \cdots \sqcap E_k \sqsubseteq \exists s.D \in \mathcal{T}$ and $s \sqsubseteq r \in \mathcal{T}$ with $\rho_{\mathsf{lhs}(\gamma),\mathcal{T}} \in (E_1 \sqcap \cdots \sqcap E_k)^{\mathcal{I}_m}$ (the case $s = r$ is similar and omitted). It follows that $A \in \mathsf{lhs}(\gamma)$ because otherwise $E_1 \sqcap \cdots \sqcap E_k \sqsubseteq \exists s.D$ would

be a target of γ since, by induction hypothesis, $\{E_1, \ldots, E_k\} \subseteq \mathsf{lhs}(\gamma)$ and $A \in (\exists s.D)^{\mathsf{sat}}$. This finishes the proof of Claim 1.

By Claim 1, as $\emptyset \not\models \gamma$, there is a conjunct of the form $\exists r.F$ in $\mathsf{rhs}(\gamma)$. Let $(\mathsf{lhs}(\alpha))^{\mathcal{T}}$ be as above and $\mathcal{I}_{D_0,\mathcal{T}}$ be the canonical model of $D_0 = \bigsqcap_{D \in (\mathsf{lhs}(\alpha))^{\mathcal{T}}} D$ and $\mathcal{T}$. As $\exists r.F \in \mathsf{rhs}(\gamma)$ and γ is $\mathcal{T}$-essential one can show in the same way as in the proof of Lemma 32 that there is an injective homomorphism from the labelled tree $T_{\exists r.F}$ corresponding to $\exists r.F$ into the restriction of $\mathcal{I}_{D_0,\mathcal{T}}$ to $\Delta^{\mathcal{I}_{D_0}}$ mapping the root of $T_{\exists r.F}$ to the root $\rho_{D_0,\mathcal{T}}$ of $\mathcal{I}_{D_0,\mathcal{T}}$. Thus, by definition of $(\mathsf{lhs}(\alpha))^{\mathcal{T}}$, there is $\beta \in \mathcal{T}$ such that $\mathcal{T} \models \mathsf{lhs}(\alpha) \sqsubseteq \mathsf{lhs}(\beta)$ and $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq \exists r.F$. By Lemma 12, $\rho_{\mathsf{lhs}(\gamma),\mathcal{T}} \in A^{\mathcal{I}_{\mathsf{lhs}(\gamma),\mathcal{T}}}$, for all $A \in \mathsf{lhs}(\beta)$. Hence, by Claim 1 and again Lemma 12, $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\gamma)$. We have shown that γ has target β and so derived a contradiction. ❑

Point (iii) is a direct consequence of Lemma 46 and the fact that all CIs in $\mathcal{H}_{add}$ are $\mathcal{T}$-essential (Lemma 38). To prove Point (iv), we first establish the following intermediate Lemmas 47 and 48.

Lemma 47 *Let $\mathcal{T}$ be rhs-primitive and let $\mathcal{H}, \gamma$ be inputs to CN-Refine. Let $\alpha_i \in \mathcal{H}_{add}$, $\beta \in \mathcal{T}$, and concept name $A \notin \mathsf{lhs}(\gamma)$ satisfy the following conditions: (a) $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\gamma)$; (b) $\mathcal{T} \models \mathsf{lhs}(\beta) \sqsubseteq A$; (c) $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i)$. Then there is some $j \leq i$ such that α_j is replaced in Line 4 of CN-Refine.*

Proof Assume $\mathcal{H}$, γ, α_i, β, and A satisfy the conditions of the lemma. If CN-Refine replaces some α_j with $j < i$ then we are done. Suppose this does not happen. Then we need to show that α_i is replaced. By Conditions (a), (b), and (c), $\mathcal{T} \models \mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_i) \sqsubseteq A$. As γ is left saturated for $\mathcal{H}$, $A \notin \mathsf{lhs}(\gamma)$ implies that $\mathcal{H} \not\models \mathsf{lhs}(\gamma) \sqsubseteq A$. So $\mathcal{H} \not\models \mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_i) \sqsubseteq A$. Then, the condition in Lines 1 and 2 of CN-Refine is satisfied and α_i is replaced. ❑

Lemma 48 *Let $\mathcal{T}$ be rhs-primitive and let $\mathcal{H}, \gamma$ be inputs to $\exists$-Refine. If γ has target $\beta \in \mathcal{T}$ and $\alpha_i \in \mathcal{H}_{add}$ satisfies $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i)$, then there is some $j \leq i$ such that α_j is replaced in Line 5 or 8 of $\exists$-Refine.*

Proof Let $\mathcal{H}$, γ, β, and α_i satisfy the conditions of the lemma. If $\exists$-Refine replaces some α_j with $j < i$ then we are done. Suppose this does not happen. We need to show that α_i is replaced. We first show that there is a concept C of the form $\exists r.F$ in $\mathsf{rhs}(\gamma)$ such that $\mathsf{lhs}(\gamma) \sqsubseteq C$ has target β. Note that if Algorithm 3 calls $\exists$-Refine then there is no concept name A such that $\mathcal{T} \models \mathsf{lhs}(\gamma) \sqsubseteq A$ and $\mathcal{H} \not\models \mathsf{lhs}(\gamma) \sqsubseteq A$ (Line 10). As γ is left saturated for $\mathcal{H}$ and $\mathcal{T}$-essential, this implies $\mathsf{N_C} \cap \mathsf{rhs}(\gamma) \subseteq \mathsf{lhs}(\gamma)$. But then any $C \in \mathsf{rhs}(\gamma) \setminus \mathsf{lhs}(\gamma)$ with $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq C$ is compound. As γ has target β it follows that $\mathsf{lhs}(\gamma) \sqsubseteq C$ has target β for some C of the form $\exists r.F$ in $\mathsf{rhs}(\gamma)$. By Line 5 of Algorithm 3, there is only one such conjunct C in $\mathsf{rhs}(\gamma)$. From $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq C$ we obtain $\mathcal{T} \models \mathsf{lhs}(\beta) \sqsubseteq C$. Since $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_i)$, we have that $\mathcal{T} \models \mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_i) \sqsubseteq C$. As γ is a positive counterexample, $\mathcal{H} \not\models \gamma$. From $\mathsf{N_C} \cap \mathsf{rhs}(\gamma) \subseteq \mathsf{lhs}(\gamma)$ we thus obtain $\mathcal{H} \not\models \mathsf{lhs}(\gamma) \sqsubseteq C$, and so, $\mathcal{H} \not\models \mathsf{lhs}(\alpha_i) \cap \mathsf{lhs}(\gamma) \sqsubseteq C$. Hence, the condition in Lines 1 and 2 of $\exists$-Refine is satisfied and α_i is replaced (in Line 5 or 8). ❑

Point (iv) above is now a direct consequence of the following lemma.

Lemma 49 *At any point in the execution of Algorithm 3, if $\alpha_j \in \mathcal{H}_{add}$ has target $\beta \in \mathcal{T}$ then $\mathsf{lhs}(\beta) \not\subseteq \mathsf{lhs}(\alpha_i)$, for all $i < j$.*

Proof The proof is by induction on the number k of iterations. For $k = 1$ the lemma is vacuously true. Assume it holds for $k = n$, $n \geq 1$. Now the algorithm modifies $\mathcal{H}_{add}$ in response to receiving a positive counterexample in iteration $k = n + 1$. We make a case distinction:

Case 1. Algorithm 3 calls CN-Refine: Let $\mathcal{H}, \gamma$ be the inputs to CN-Refine. Assume first that the condition in Lines 1 and 2 is not satisfied. Then CN-Refine appends the result of concept saturating γ for $\mathcal{T}$ to $\mathcal{H}_{add}$. Call this CI γ'. Suppose that the lemma fails to hold. This can only happen if γ' has a target $\beta \in \mathcal{T}$ and there is $\alpha_i \in \mathcal{H}_{add}$ such that $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i)$. Then, since $\mathsf{lhs}(\gamma') = \mathsf{lhs}(\gamma)$, we have that $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\gamma)$ and, since $\mathsf{rhs}(\gamma') \subseteq \mathsf{N_C}$, there is a concept name $A \notin \mathsf{lhs}(\gamma)$ such that $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq A$. So $\mathcal{T} \models \mathsf{lhs}(\beta) \sqsubseteq A$. Then Lemma 47 applies to $\mathcal{H}$, γ, α_i, β and A which contradicts the assumption that CN-Refine did not replace any $\alpha_j \in \mathcal{H}_{add}$, $j \leq i$.

Now assume that the condition in Lines 1 and 2 is satisfied. Suppose that the lemma fails to hold. This can only happen if there are $\alpha_i, \alpha_j \in \mathcal{H}_{add}$ with $i < j$ such that either (a) α_i is replaced by α_i', $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i')$ and α_j has target β; or (b) α_j is replaced by α_j', α_j' has target β and $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i)$. In case (a), from $\mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_i) = \mathsf{lhs}(\alpha_i')$, we obtain $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_i)$. Thus, $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i)$. This contradicts the induction hypothesis. Now assume case (b). Since α_j' has target β, we obtain:

1. $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_j')$; and
2. as $\mathsf{rhs}(\alpha_j') \subseteq \mathsf{N_C}$, there is $A \in \mathsf{N_C}$ with $A \in \mathsf{rhs}(\alpha_j') \setminus \mathsf{lhs}(\alpha_j')$ and $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq A$.

Since $\mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_j) = \mathsf{lhs}(\alpha_j')$, it follows from Point 1 that $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_j)$ and $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\gamma)$. From $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq A$ we obtain $\mathcal{T} \models \mathsf{lhs}(\beta) \sqsubseteq A$. If $A \in \mathsf{rhs}(\alpha_j') \setminus \mathsf{lhs}(\alpha_j')$ then either $A \in \mathsf{rhs}(\alpha_j) \setminus \mathsf{lhs}(\alpha_j)$ or $A \notin \mathsf{lhs}(\gamma)$. So either α_j has target β or $A \notin \mathsf{lhs}(\gamma)$. α_j does not have target β as this would contradict the induction hypothesis. Thus, $A \notin \mathsf{lhs}(\gamma)$ and the conditions of Lemma 47 are satisfied by $\mathcal{H}$, γ, α_i, β, and A. Thus, some $\alpha_{i'}$ with $i' \leq i$ is replaced which contradicts the assumption that α_j is replaced.

Case 2. Algorithm 3 calls $\exists$-Refine: Let $\mathcal{H}, \gamma$ be the inputs to $\exists$-Refine. Assume first that the condition in Lines 1 and 2 is not satisfied. Then $\exists$-Refine appends γ to $\mathcal{H}_{add}$. Suppose the lemma fails to hold. This can only happen if γ has a target $\beta \in \mathcal{T}$ and there is $\alpha_i \in \mathcal{H}_{add}$ such that $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i)$. By Lemma 48, this contradicts the assumption that $\exists$-Refine did not replace any $\alpha_j \in \mathcal{H}_{add}$, $j \leq i$.

Assume now that the condition in Lines 1 and 2 is satisfied. Suppose that the lemma fails to hold. This can only happen if there are $\alpha_i, \alpha_j \in \mathcal{H}_{add}$ with $i < j$ such that either (a) α_i is replaced by α_i', $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i')$ and α_j has target β; or (b) α_j is replaced by α_j', α_j' has target β and $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i)$. For case (a) we argue as above: from $\mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_i) = \mathsf{lhs}(\alpha_i')$, we obtain $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_i)$. Thus, $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_i)$, which contradicts the induction hypothesis. Now assume case (b). As α_j' has target β, we obtain the following:

1. $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha'_j)$; and

2. there is $D \in \mathsf{rhs}(\alpha'_j) \setminus \mathsf{lhs}(\alpha'_j)$ and $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq D$.

Since $\mathsf{lhs}(\gamma) \cap \mathsf{lhs}(\alpha_j) = \mathsf{lhs}(\alpha'_j)$, it follows from Point 1 that $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\alpha_j)$ and $\mathsf{lhs}(\beta) \subseteq \mathsf{lhs}(\gamma)$. Recall that if Algorithm 3 calls $\exists$-Refine then there is no $A \in \mathsf{N_C}$ such that $\mathcal{T} \models \mathsf{lhs}(\gamma) \sqsubseteq A$ and $\mathcal{H} \not\models \mathsf{lhs}(\gamma) \sqsubseteq A$. So $\mathsf{N_C} \cap \mathsf{rhs}(\gamma) \subseteq \mathsf{lhs}(\gamma)$ (by left saturation of γ for $\mathcal{H}$). Assume $D \in \mathsf{N_C}$. Since $D \in \mathsf{rhs}(\alpha'_j) \setminus \mathsf{lhs}(\alpha'_j)$ (Point 2), it follows that $D \notin \mathsf{lhs}(\alpha_j)$. As $\mathsf{lhs}(\alpha'_j) \subseteq \mathsf{lhs}(\alpha_j)$, we have that $D \in \mathsf{rhs}(\alpha_j)$. So $D \in \mathsf{rhs}(\alpha_j) \setminus \mathsf{lhs}(\alpha_j)$. This means that α_j has target β, which contradicts the induction hypothesis. Otherwise, D is of the form $\exists r.F$. Then, either $D \in \mathsf{rhs}(\gamma)$ or there is $D' \in \mathsf{rhs}(\alpha_j)$ such that $\emptyset \models D \sqsubseteq D'$. In the latter case, $D' \in \mathsf{rhs}(\alpha_j) \setminus \mathsf{lhs}(\alpha_j)$ and $\emptyset \models \mathsf{rhs}(\beta)^{\mathsf{sat}} \sqsubseteq D'$, so α_j has target β, which contradicts the induction hypothesis. In the former case, γ has target β. Then $\mathcal{H}$, γ, and α_i satisfy the conditions of Lemma 48. Thus, some $\alpha_{i'}$ with $i' \leq i$ is replaced which contradicts the assumption that α_j is replaced. ❑

We have proved the main result of this section.

Theorem 50 *DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBoxes are polynomial query learnable using membership and equivalence queries. Moreover, DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBoxes without inverse roles can be learned in polynomial time using membership and equivalence queries.*

Proof Polynomial query learnability of DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBoxes follows from Lemma 41 and the analysis of the number of membership queries in Lemmas 39 and 40, see the proof of Theorem 34. For the second part observe that the only potentially costly steps are entailment checks of the form $\mathcal{H} \models \alpha$, where $\mathcal{H}$ is a DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ TBox and α a DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ CI, both without inverse roles. Then both $\mathcal{H}$ and α are in $\mathcal{EL}$ with role inclusions for which entailment is known to be in PTime (Baader et al., 2005). ❑

5. Learning $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes

We study polynomial learnability of TBoxes in the restriction $\mathcal{EL}_{\mathsf{lhs}}$ of $\mathcal{EL}$ in which only concept names are allowed on the right-hand side of CIs. We assume that CIs used in membership queries and in equivalence queries and those returned as counterexamples are also of this restricted form and show that under this assumption $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes can be learned in polynomial time. As in the previous section, our learning algorithm is an extension of the polynomial time algorithm for learning propositional Horn theories presented by Angluin et al. (1992) and Arias and Balcázar (2011).

There is a certain similarity between the learning algorithm of this section and the DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ learning algorithm introduced in Section 4. In both cases the left-hand side of inclusions can contain complex concept expressions, which, unless addressed, might lead to several counterexamples with unnecessarily strong left-hand sides targeting the same inclusion in the target TBox. In Algorithm 3 storing multiple such counterexamples in $\mathcal{H}_{add}$ is prevented by taking the intersection of the set of conjuncts of the left-hand sides. To deal with the more complex left-hand sides of inclusions in $\mathcal{EL}_{\mathsf{lhs}}$, a more sophisticated way of 'taking the intersection' of concept expressions is required. To define it, we identify concept

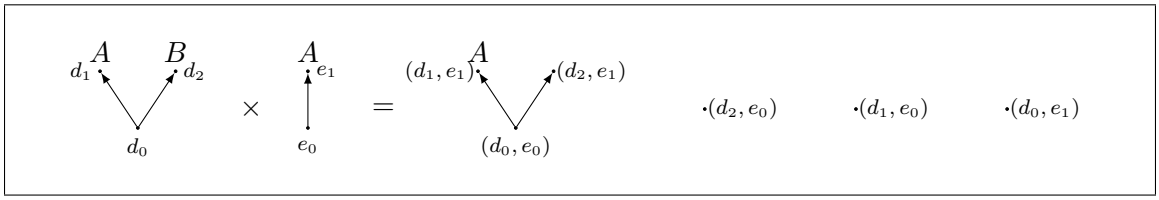

Figure 11: Illustration to Example 52.

expressions with tree-shaped interpretations and then take their product. Products have also been employed in the construction of least common subsumers (Baader et al., 1999).

In detail, we say that an interpretation $\mathcal{I}$ is a *ditree interpretation* if the directed graph $(\Delta^{\mathcal{I}}, E)$ with $E = \bigcup_{r \in \mathsf{N_R}} r^{\mathcal{I}}$ is a directed tree and $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$ for all distinct $r, s \in \mathsf{N_R}$. We denote the root of a ditree interpretation $\mathcal{I}$ with $\rho_{\mathcal{I}}$. The interpretation $\mathcal{I}_C$ corresponding to an $\mathcal{EL}$ concept expression C is a ditree interpretation with root ρ_C. Conversely, every ditree interpretation $\mathcal{I}$ can be viewed as an $\mathcal{EL}$ concept expression $C_{\mathcal{I}}$ in the same way as any labelled tree T with edge labels that are role names (rather than arbitrary roles) can be seen as an $\mathcal{EL}$ concept expression.

An interpretation $\mathcal{I}$ is a *$\mathcal{T}$-countermodel* for a given $\mathcal{EL}_{\mathsf{lhs}}$ TBox $\mathcal{T}$ if $\mathcal{I} \not\models \mathcal{T}$. Notice that for any $\mathcal{EL}_{\mathsf{lhs}}$ inclusion $C \sqsubseteq A$ with $\mathcal{T} \models C \sqsubseteq A$ and $\emptyset \not\models C \sqsubseteq A$ the interpretation $\mathcal{I}_C$ is a $\mathcal{T}$-countermodel. Indeed, by construction of $\mathcal{I}_C$, we have $\rho_C \in C^{\mathcal{I}_C}$ and, as $\emptyset \not\models C \sqsubseteq A$, we have $\rho_C \notin A^{\mathcal{I}_C}$. So $\mathcal{I}_C \not\models C \sqsubseteq A$ and, as $\mathcal{T} \models C \sqsubseteq A$, we have $\mathcal{I}_C \not\models \mathcal{T}$. Conversely, given a $\mathcal{T}$-countermodel $\mathcal{I}$, a learning algorithm can construct in polynomial time in $|\Sigma_{\mathcal{T}}|$ all inclusions of the form $C_{\mathcal{I}} \sqsubseteq A$, where A is a concept name, such that $\mathcal{T} \models C_{\mathcal{I}} \sqsubseteq A$ by posing membership queries to the oracle. Thus a learning algorithm can use inclusions and $\mathcal{T}$-countermodels interchangeably. We prefer working with interpretations as we can then use the notion of products to define the 'intersection of concept expressions' and the results of Section 2 linking homomorphisms with entailment in a direct way.

The *product* of two interpretations $\mathcal{I}$ and $\mathcal{J}$ is the interpretation $\mathcal{I} \times \mathcal{J}$ with

$$\begin{array}{rcl} \Delta^{\mathcal{I}\times\mathcal{J}} & = & \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}} \\ A^{\mathcal{I}\times\mathcal{J}} & = & \{(d, e) \mid d \in A^{\mathcal{I}}, e \in A^{\mathcal{J}}\} \\ r^{\mathcal{I}\times\mathcal{J}} & = & \{((d, e), (d', e')) \mid (d, d') \in r^{\mathcal{I}}, (e, e') \in r^{\mathcal{J}}\} \end{array}$$

Products preserve the membership in $\mathcal{EL}$ concept expressions (Lutz et al., 2011):

Lemma 51 *For all interpretations $\mathcal{I}$ and $\mathcal{J}$, all $d \in \Delta^{\mathcal{I}}$ and $e \in \Delta^{\mathcal{J}}$, and for all $\mathcal{EL}$ concept expressions C the following holds: $d \in C^{\mathcal{I}}$ and $e \in C^{\mathcal{J}}$ if, and only if, $(d, e) \in C^{\mathcal{I}\times\mathcal{J}}$.*

One can easily show that the product of ditree interpretations is a disjoint union of ditree interpretations. If $\mathcal{I}$ and $\mathcal{J}$ are ditree interpretations, we denote by $\mathcal{I} \times_\rho \mathcal{J}$ the maximal ditree interpretation that is a subinterpretation of $\mathcal{I} \times \mathcal{J}$ and contains $(\rho_{\mathcal{I}}, \rho_{\mathcal{J}})$.

Example 52 *Figure 11 depicts the product of the ditree interpretations $\mathcal{I}$ with root d_0 and $\mathcal{J}$ with root e_0. The ditree interpretation $\mathcal{I} \times_\rho \mathcal{J}$ has root (d_0, e_0) and does not contain the nodes (d_2, e_0), (d_1, e_0) and (d_0, e_1) from $\mathcal{I} \times \mathcal{J}$.*

Observe that the product $\mathcal{I}_C \times \mathcal{I}_D$ of concept expressions $C = A_1 \sqcap \cdots \sqcap A_n$ and $D = B_1 \sqcap \cdots \sqcap B_m$, where $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$ are concept names, coincides with the interpretation $\mathcal{I}_E$, where E is the conjunction of all concept names in $\{A_1, \ldots, A_n\} \cap \{B_1, \ldots, B_m\}$. Thus, products can be seen as a generalisation of taking the intersection of the concept names from the left-hand side of DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ concept inclusions used in Section 4.

We will now describe a class of $\mathcal{T}$-countermodels that are in a sense minimal and central to our learning algorithm. Let $\mathcal{T}$ be the $\mathcal{EL}_{\mathsf{lhs}}$ TBox to be learned, and assume that its signature $\Sigma_{\mathcal{T}}$ is known to the learner. For a ditree interpretation $\mathcal{I}$, we use $\mathcal{I}|^-_\rho$ to denote the interpretation obtained from $\mathcal{I}$ by removing the root $\rho_{\mathcal{I}}$ of $\mathcal{I}$. For any $d \in \Delta^{\mathcal{I}} \setminus \{\rho_{\mathcal{I}}\}$, we use $\mathcal{I}|^-_{d\downarrow}$ to denote $\mathcal{I}$ with the subtree rooted at d removed. A $\mathcal{T}$-countermodel is *essential* if the following conditions are satisfied:

1. $\mathcal{I}|^-_\rho \models \mathcal{T}$;
2. $\mathcal{I}|^-_{d\downarrow} \models \mathcal{T}$ for all $d \in \Delta^{\mathcal{I}} \setminus \{\rho_{\mathcal{I}}\}$.

Intuitively, Condition 1 states that $\mathcal{I}$ contradicts $\mathcal{T}$ only at the root, that is, the only reason for why $\mathcal{I}$ does not satisfy $\mathcal{T}$ is that for at least one CI $C \sqsubseteq A \in \mathcal{T}$, we have that $\rho_{\mathcal{I}} \in C^{\mathcal{I}}$ and $\rho_{\mathcal{I}} \notin A^{\mathcal{I}}$. Condition 2 is a minimality condition which states that for any such $C \sqsubseteq A \in \mathcal{T}$, $\rho_{\mathcal{I}}$ is no longer in $C^{\mathcal{I}}$ if we remove any node from $\mathcal{I}$. Example 61 at the end of this section shows that working with essential $\mathcal{T}$-countermodels is needed for our learning algorithm to be in polynomial time.

The algorithm for learning $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes is given as Algorithm 6. It maintains an ordered list $\mathfrak{I}$ of ditree interpretations that intuitively represents the TBox $\mathcal{H}$ constructed in Line 13. In Line 6 we write $\mathcal{I} \to_\rho \mathcal{J}$ if there is a homomorphism from a ditree interpretation $\mathcal{I}$ to a ditree interpretation $\mathcal{J}$ mapping $\rho_{\mathcal{I}}$ to $\rho_{\mathcal{J}}$. $\mathcal{I} \not\to_\rho \mathcal{J}$ denotes that no such homomorphism exists. By Lemma 10, $\mathcal{I} \to_\rho \mathcal{J}$ iff $\emptyset \models C_{\mathcal{J}} \sqsubseteq C_{\mathcal{I}}$ which can be checked in polynomial time in the size of $\mathcal{I}$ and $\mathcal{J}$. In Line 8, we write $\mathcal{J}' \subseteq \mathcal{I} \times_\rho \mathcal{J}$ as shorthand for the condition that $\mathcal{J}'$ is a subinterpretation of $\mathcal{I} \times_\rho \mathcal{J}$ that is obtained from $\mathcal{I} \times_\rho \mathcal{J}$ by removing subtrees. Note that the assumption in Line 4 that a *positive* counterexample is returned is justified by the construction of $\mathcal{H}$ in Lines 2 and 13, which ensures that, at all times, $\mathcal{T} \models \mathcal{H}$.

We now provide additional details on how to realise lines 5, 8 and 13. Line 13 is the easiest: we simply use membership queries '$\mathcal{T} \models C_{\mathcal{I}} \sqsubseteq A$?' with $\mathcal{I} \in \mathfrak{I}$ and $A \in \Sigma_{\mathcal{T}}$ to find all CIs $C_{\mathcal{I}} \sqsubseteq A$ entailed by $\mathcal{T}$. We will later show that the length of $\mathfrak{I}$ is bounded polynomially in $|\mathcal{T}|$ and that each interpretation in $\mathfrak{I}$ is replaced only polynomially many times, therefore polynomially many membership queries suffice. Lines 5 and 8 are addressed by Lemmas 53 and 54 below.

Lemma 53 *Given a positive counterexample $C \sqsubseteq A$ for $\mathcal{T}$ relative to $\mathcal{H}$, one can construct an essential $\mathcal{T}$-countermodel $\mathcal{I}$ with $\mathcal{I} \models \mathcal{H}$ using only polynomially many membership queries in $|\mathcal{T}| + |C|$.*

Proof Let $C \sqsubseteq A$ be a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}$. Let $\mathcal{I}_C$ be the ditree interpretation of C. First observe that $\mathcal{I}_C \not\models \mathcal{T}$: since $\mathcal{H} \not\models C \sqsubseteq A$, we know that A does not occur as a top-level conjunct in C. Consequently, $\rho_C \in C^{\mathcal{I}_C} \setminus A^{\mathcal{I}_C}$ and thus $\mathcal{I}_C \not\models \mathcal{T}$.

We construct an essential $\mathcal{T}$-countermodel $\mathcal{I}$ with $\mathcal{I} \models \mathcal{H}$ by applying the following rules to $\mathcal{I} := \mathcal{I}_C$.

Algorithm 6 The learning algorithm for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes

Input: $\mathcal{EL}_{\mathsf{lhs}}$ TBox $\mathcal{T}$ given to the oracle; $\Sigma_{\mathcal{T}}$ given to the learner.

Output: TBox $\mathcal{H}$, computed by the learner, such that $\mathcal{T} \equiv \mathcal{H}$.

```
Set 𝔍 to the empty list (of ditree interpretations)
Set ℋ = ∅
while ℋ ⊭≡ 𝒯 do
    Let C ⊑ A be the returned positive counterexample for 𝒯 relative to ℋ
    Find an essential 𝒯-countermodel ℐ with ℐ ⊨ ℋ
    if there is a 𝒥 ∈ 𝔍 such that 𝒥 ↛_ρ (ℐ ×_ρ 𝒥) and ℐ ×_ρ 𝒥 ⊭ 𝒯 then
        Let 𝒥 be the first such element of 𝔍
        Find an essential 𝒯-countermodel 𝒥' ⊆ ℐ ×_ρ 𝒥
        Replace 𝒥 in 𝔍 with 𝒥'
    else
        Append ℐ to 𝔍
    end if
    Construct ℋ = {C_ℐ ⊑ A | ℐ ∈ 𝔍, A a concept name in Σ_𝒯, 𝒯 ⊨ C_ℐ ⊑ A}
end while
return ℋ
```

1. Saturate $\mathcal{I}$ by exhaustively applying the CIs from $\mathcal{H}$ as rules: if $D \sqsubseteq B \in \mathcal{H}$ and $d \in D^{\mathcal{I}}$, then add d to $B^{\mathcal{I}}$.

2. Replace $\mathcal{I}$ by a minimal subtree of $\mathcal{I}$ refuting $\mathcal{T}$ to address Condition 1 of essential $\mathcal{T}$-countermodels. To describe how this can be achieved using membership queries denote for $d \in \Delta^{\mathcal{I}}$ by $\mathcal{I}|_d$ the ditree interpretation obtained from $\mathcal{I}$ by taking the subtree of $\mathcal{I}$ rooted in d. Now check using membership queries for any $d \in \Delta^{\mathcal{I}} \setminus \{\rho_{\mathcal{I}}\}$ and concept name B whether $\mathcal{T} \models C_{\mathcal{I}|_d} \sqsubseteq B$. Then replace $\mathcal{I}$ by any $\mathcal{I}|_d$ such that there exists a B with $\mathcal{T} \models C_{\mathcal{I}|_d} \sqsubseteq B$ and $d \notin B^{\mathcal{I}|_d}$ but there does not exist a d' in $\Delta^{\mathcal{I}|_d}$ and a B' with $\mathcal{T} \models C_{\mathcal{I}|_{d'}} \sqsubseteq B'$ and $d' \notin B^{\mathcal{I}|_{d'}}$. If no such d and B exist, then $\mathcal{I}$ is not replaced.

3. Exhaustively remove subtrees from $\mathcal{I}$ until Condition 2 of essential $\mathcal{T}$-countermodels is also satisfied: if $\mathcal{I}|^-_{d\downarrow} \not\models \mathcal{T}$, then replace $\mathcal{I}$ by $\mathcal{I}|^-_{d\downarrow}$. This can again be achieved using the membership queries $\mathcal{T} \models C_{\mathcal{I}|^-_{d\downarrow}} \sqsubseteq B$ for B a concept name.

Now we show that the interpretation $\mathcal{J}$ constructed above has the required properties. First observe that $\mathcal{J} \models \mathcal{H}$: clearly, the interpretation $\mathcal{I}$ constructed in Step 1 is a model of $\mathcal{H}$. As taking subtrees and removing subtrees from $\mathcal{I}$ preserves being a model of $\mathcal{H}$, we conclude that $\mathcal{J} \models \mathcal{H}$. Next we show that $\mathcal{J} \not\models \mathcal{T}$: the interpretation $\mathcal{I}$ constructed in Step 1 is not a model of $\mathcal{T}$. In fact, we can use $C_{\mathcal{I}} \sqsubseteq A$ as a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}$ instead of $C \sqsubseteq A$. Observe that $\emptyset \models C_{\mathcal{I}} \sqsubseteq C$, and thus $\mathcal{T} \models C \sqsubseteq A$ implies $\mathcal{T} \models C_{\mathcal{I}} \sqsubseteq A$. On the other hand, $\rho_{\mathcal{I}} \in B^{\mathcal{I}}$ implies $\mathcal{H} \models C \sqsubseteq B$ for all concept names B. Consequently and since $\mathcal{H} \not\models C \sqsubseteq A$, we have $\rho_{\mathcal{I}} \notin A^{\mathcal{I}}$. Thus $\mathcal{I} \not\models \mathcal{T}$. By construction, Steps 2 and 3 preserve the condition that $\mathcal{I}$ is not a model of $\mathcal{T}$ and so $\mathcal{J} \not\models \mathcal{T}$. It remains to argue that $\mathcal{J}$

satisfies Conditions 1 and 2 for essential $\mathcal{T}$-countermodels for $\mathcal{H}$. But Condition 1 is ensured by Step 2 and Condition 2 is ensured by Step 3, respectively. ❑

Lemma 54 *Given essential $\mathcal{T}$-countermodels $\mathcal{I}$ and $\mathcal{J}$ with $\mathcal{I} \times_\rho \mathcal{J} \not\models \mathcal{T}$, one can construct an essential $\mathcal{T}$-countermodel $\mathcal{J}' \subseteq \mathcal{I} \times_\rho \mathcal{J}$ using only polynomially many membership queries in $|\mathcal{T}| + |\mathcal{I}| + |\mathcal{J}|$.*

Proof Let $\mathcal{I}$ and $\mathcal{J}$ be essential $\mathcal{T}$-countermodels with $\mathcal{I} \times_\rho \mathcal{J} \not\models \mathcal{T}$. Obtain the interpretation $\mathcal{J}'$ from $\mathcal{I} \times_\rho \mathcal{J}$ by exhaustively applying Rule 3 from the proof of Lemma 53. As argued above, applying Rule 3 can be implemented using membership queries and $\mathcal{J}'$ is a $\mathcal{T}$-countermodel. Thus, it remains to argue that it satisfies Conditions 1 and 2 for $\mathcal{T}$-essential countermodels. For Condition 1, we have to show that $\mathcal{J}'|_\rho^- \models \mathcal{T}$. We know that $\mathcal{I}|_\rho^- \models \mathcal{T}$ and $\mathcal{J}|_\rho^- \models \mathcal{T}$. Thus, by Lemma 51, $\mathcal{I}|_\rho^- \times \mathcal{J}|_\rho^- \models \mathcal{T}$. Now $\mathcal{J}'|_\rho^-$ is obtained from $\mathcal{I}|_\rho^- \times \mathcal{J}|_\rho^-$ by removing subtrees and removing subtrees preserves being a model of an $\mathcal{EL}_{\mathsf{lhs}}$ TBox. Thus, $\mathcal{J}'|_\rho^- \models \mathcal{T}$. For Condition 2, we have to show that $\mathcal{J}'|_{d\downarrow}^- \models \mathcal{T}$ for all $d \in \Delta^{\mathcal{J}'} \setminus \{\rho_{\mathcal{J}'}\}$. But if this is not the case, then the subtree rooted at d would have been removed during the construction of $\mathcal{J}'$ from $\mathcal{I} \times_\rho \mathcal{J}$ using Rule 3. ❑

If Algorithm 6 terminates, then it obviously returns a TBox $\mathcal{H}$ that is equivalent to the target TBox $\mathcal{T}$. It thus remains to prove that the algorithm terminates after polynomially many steps, which is a consequence of the following lemma.

Lemma 55 *Let $\mathfrak{I}$ be a list computed at some point of an execution of Algorithm 6. Then (i) the length of $\mathfrak{I}$ is bounded by the number of CIs in $\mathcal{T}$ and (ii) each interpretation in each position of $\mathfrak{I}$ is replaced only $|\mathcal{T}| + |\mathcal{T}|^2$ often with a new interpretation.*

The rest of this section is devoted to proving Lemma 55. For easy reference, assume that at each point of the execution of the algorithm, $\mathfrak{I}$ has the form $\mathcal{I}_0, \ldots, \mathcal{I}_k$ for some $k \geq 0$. To establish Point (i) of Lemma 55, we closely follow the argument given by Angluin et al. (1992) and show that

(iii) for every $\mathcal{I}_i$, there is a $D_i \sqsubseteq A_i \in \mathcal{T}$ with $\mathcal{I}_i \not\models D_i \sqsubseteq A_i$ and

(iv) if $i \neq j$, then $D_i \sqsubseteq A_i$ and $D_j \sqsubseteq A_j$ are not identical.

In fact, Point (iii) is immediate since whenever a new $\mathcal{I}_i$ is added to $\mathfrak{I}$ in the algorithm, then $\mathcal{I}_i$ is a $\mathcal{T}$-countermodel. To prove Point (iv), we first establish the intermediate Lemma 56 below. For a ditree interpretation $\mathcal{I}$ and a CI $C \sqsubseteq A$, we write $\mathcal{I} \models^\rho C \sqsubseteq A$ if $\rho_\mathcal{I} \notin C^\mathcal{I}$ or $\rho_\mathcal{I} \in A^\mathcal{I}$; that is, the CI $C \sqsubseteq A$ is satisfied at the root of $\mathcal{I}$, but not necessarily at other points in $\mathcal{I}$. It is easy to see that if some interpretation $\mathcal{I}$ is a $\mathcal{T}$-countermodel, then there is $C \sqsubseteq A \in \mathcal{T}$ such that $\mathcal{I} \not\models^\rho C \sqsubseteq A$.

The following lemma shows under which conditions Algorithm 6 replaces an interpretation in the list $\mathfrak{I}$.

Lemma 56 *If the interpretation $\mathcal{I}$ constructed in Line 5 of Algorithm 6 satisfies $\mathcal{I} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$ for some j, then $\mathcal{J} = \mathcal{I}_i$ is replaced with $\mathcal{J}'$ in Line 9 for some $i \leq j$.*

Proof Assume that the interpretation $\mathcal{I}$ constructed in Line 5 of Algorithm 6 satisfies $\mathcal{I} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and that there is some j with $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$. If there is some $i < j$ such that $\mathcal{I}_i \not\to_\rho (\mathcal{I} \times_\rho \mathcal{I}_i)$ and $\mathcal{I} \times_\rho \mathcal{I}_i \not\models \mathcal{T}$, then $\mathcal{J} = \mathcal{I}_{i'}$ will be replaced with $\mathcal{J}'$ in Line 9 for some $i' \leq i$ and we are done. Thus assume that there is no such i. We aim to show that $\mathcal{J} = \mathcal{I}_j$ is replaced with $\mathcal{J}'$ in Line 9. To this end, it suffices to prove that $\mathcal{I}_j \not\to_\rho (\mathcal{I} \times_\rho \mathcal{I}_j)$ and $\mathcal{I} \times_\rho \mathcal{I}_j \not\models \mathcal{T}$. The latter is a consequence of $\mathcal{I} \not\models^\rho C \sqsubseteq A$ and $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$.

Assume to the contrary of what we have to show that $\mathcal{I}_j \to_\rho (\mathcal{I} \times_\rho \mathcal{I}_j)$. We establish a contradiction against $\mathcal{I} \models \mathcal{H}$ (which holds by construction of $\mathcal{I}$ in the algorithm) by showing that

1. $\mathcal{I} \not\models^\rho C_{\mathcal{I}_j} \sqsubseteq A$ and
2. $C_{\mathcal{I}_j} \sqsubseteq A \in \mathcal{H}$.

For Point 1, $\mathcal{I}_j \to_\rho (\mathcal{I} \times_\rho \mathcal{I}_j)$ and $\rho_{\mathcal{I}_j} \in (C_{\mathcal{I}_j})^{\mathcal{I}_j}$ imply $\rho_{\mathcal{I} \times_\rho \mathcal{I}_j} \in (C_{\mathcal{I}_j})^{\mathcal{I} \times_\rho \mathcal{I}_j}$, which gives $\rho_{\mathcal{I}} \in (C_{\mathcal{I}_j})^{\mathcal{I}}$, by Lemma 51. It remains to observe that $\mathcal{I} \not\models^\rho C \sqsubseteq A$ implies $\rho_{\mathcal{I}} \notin A^{\mathcal{I}}$.

In view of the construction of $\mathcal{H}$ in the algorithm, Point 2 can be established by showing that $\mathcal{T} \models C_{\mathcal{I}_j} \sqsubseteq A$. Since $C \sqsubseteq A \in \mathcal{T}$, it suffices to prove that $\emptyset \models C_{\mathcal{I}_j} \sqsubseteq C$. This, however, is an immediate consequence of the fact that $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$ and the definition of $C_{\mathcal{I}_j}$. ❑

Now, Point (iv) above is a consequence of the following.

Lemma 57 *At any time of the algorithm execution, the following condition holds: if $\mathcal{I}_i \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $j < i$, then $\rho_{\mathcal{I}_j} \notin C^{\mathcal{I}_j}$.*

Proof We prove the invariant formulated in Lemma 57 by induction on the number of iterations of the while loop. Clearly, the invariant is satisfied before the loop is entered. We now consider the two places where $\mathfrak{I}$ is modified, that is, Line 9 and Line 11, starting with the latter.

In Line 11, $\mathcal{I}$ is appended to $\mathfrak{I}$. Assume that $\mathcal{I} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$. We have to show that, before $\mathcal{I}$ was added to $\mathfrak{I}$, there was no $\mathcal{I}_i \in \mathfrak{I}$ with $\rho_{\mathcal{I}_i} \in C^{\mathcal{I}_i}$. This, however, is immediate by Lemma 56.

Now assume that $\mathcal{J}$ was replaced in Line 9 with $\mathcal{J}'$. We have to show two properties:

1. If $\mathcal{J}' = \mathcal{I}_i \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $j < i$, then $\rho_{\mathcal{I}_j} \notin C^{\mathcal{I}_j}$.

   Assume to the contrary that $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$. Since $\mathcal{J}'$ is obtained from $\mathcal{I} \times \mathcal{J}$ by removing subtrees (see Lemma 54), $\mathcal{J}' \not\models^\rho C \sqsubseteq A$ implies $\mathcal{I} \times \mathcal{J} \not\models^\rho C \sqsubseteq A$. Consequently, $\mathcal{I} \not\models^\rho C \sqsubseteq A$ or $\mathcal{J} \not\models^\rho C \sqsubseteq A$. The former and $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$ yields $i \leq j$ by Lemma 56, in contradiction to $j < i$. In the latter case, since $\mathcal{I}_i = \mathcal{J}$ before the replacement of $\mathcal{J}$ with $\mathcal{J}'$, we have a contradiction against the induction hypothesis.

2. If $\mathcal{J}' = \mathcal{I}_j$ and $\mathcal{I}_i \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ with $i > j$, then $\rho_{\mathcal{I}_j} \notin C^{\mathcal{I}_j}$.

   Assume to the contrary that $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$. Since $\mathcal{J}'$ is obtained from $\mathcal{I} \times_\rho \mathcal{J}$ by removing subtrees, we then have $\rho_{\mathcal{I} \times_\rho \mathcal{J}} \in C^{\mathcal{I} \times_\rho \mathcal{J}}$, thus $\rho_{\mathcal{J}} \in C^{\mathcal{J}}$. Since $\mathcal{I}_j = \mathcal{J}$ before the replacement of $\mathcal{J}$ with $\mathcal{J}'$, we have a contradiction against the induction hypothesis.

❑

We now turn towards proving Point (ii) of Lemma 55. It is a consequence of Lemma 59 below.

Lemma 58 *If $\mathcal{I}$ is an essential $\mathcal{T}$-countermodel, then $|\Delta^{\mathcal{I}}| \leq |\mathcal{T}|$.*

Proof Let $\mathcal{I}$ be an essential $\mathcal{T}$-countermodel. Then $\mathcal{I} \not\models \mathcal{T}$, but $\mathcal{I}|_\rho^- \models \mathcal{T}$. It follows that there is a $C \sqsubseteq A \in \mathcal{T}$ such that $\rho_{\mathcal{I}} \in C^{\mathcal{I}} \setminus A^{\mathcal{I}}$. By Lemma 8, there is a homomorphism h from $\mathcal{I}_C$ to $\mathcal{I}$ mapping $\rho_{\mathcal{I}_C}$ to $\rho_{\mathcal{I}}$. We show that $|\Delta^{\mathcal{I}}| \leq |C|$, from which $|\Delta^{\mathcal{I}}| \leq |\mathcal{T}|$ follows. It suffices to show that h is surjective. Assume that this is not the case and let $d \in \Delta^{\mathcal{I}}$ be outside the range of h. Then h is a homomorphism from $\mathcal{I}_C$ to $\mathcal{J} := \mathcal{I}|_{d\downarrow}^-$. Therefore, $\rho^{\mathcal{J}} \in C^{\mathcal{J}}$ by Lemma 8, which implies $\mathcal{J} \not\models C \sqsubseteq A$. But $\mathcal{J} \not\models C \sqsubseteq A$ contradicts the assumption that $\mathcal{I}$ is an essential $\mathcal{T}$-countermodel as it violates Condition 2 of being an essential $\mathcal{T}$-countermodel. ❏

Lemma 59 *Let $\mathcal{I}_0, \ldots, \mathcal{I}_n$ be a list of interpretations such that $\mathcal{I}_{i+1}$ replaces $\mathcal{I}_i$ in Line 9 for all $i < n$. Then $n \leq |\mathcal{T}| + |\mathcal{T}|^2$.*

Proof Let $\mathcal{I}$ and $\mathcal{J}$ be ditree interpretations. We set $\mathcal{I} \leq_\rho \mathcal{J}$ if $\rho_{\mathcal{I}} \in A^{\mathcal{I}}$ implies $\rho_{\mathcal{J}} \in A^{\mathcal{J}}$ for all concept names A. We first show that for every $i < n$ either

(a) $\mathcal{I}_i \not\leq_\rho \mathcal{I}_{i+1}$ or

(b) $\mathcal{I}_{i+1} \to_\rho \mathcal{I}_i$ via a surjective homomorphism.

For a proof by contradiction assume that there is $i < n$ such that neither (a) nor (b) holds. Since $\mathcal{I}_{i+1}$ is obtained from some $\mathcal{I} \times_\rho \mathcal{I}_i$ by removing subtrees and $(\mathcal{I} \times_\rho \mathcal{I}_i) \to_\rho \mathcal{I}_i$ we obtain that $\mathcal{I}_{i+1} \to_\rho \mathcal{I}_i$. Since $\mathcal{I}_{i+1}$ is an essential $\mathcal{T}$-countermodel, there is a $C \sqsubseteq A \in \mathcal{T}$ such that $\mathcal{I}_{i+1} \not\models^\rho C \sqsubseteq A$. Let $\mathcal{J}$ be the subinterpretation of $\mathcal{I}_i$ determined by the range of the homomorphism h from $\mathcal{I}_{i+1}$ to $\mathcal{I}_i$ mapping $\rho_{\mathcal{I}_{i+1}}$ to $\rho_{\mathcal{I}_i}$. By Lemma 8, $\rho_{\mathcal{J}} \in C^{\mathcal{J}}$ and so, since $\rho_{\mathcal{J}} \notin A^{\mathcal{J}}$ because (a) does not hold, $\mathcal{J} \not\models^\rho C \sqsubseteq A$. $\mathcal{I}_i$ is an essential $\mathcal{T}$-countermodel and so $\mathcal{J} = \mathcal{I}$. But then h is surjective and we have derived a contradiction.

In addition to the property stated above, we also have for all $i < n$:

(c) $\mathcal{I}_{i+1} \leq_\rho \mathcal{I}_i$ and

(d) $\mathcal{I}_i \not\to_\rho \mathcal{I}_{i+1}$.

It follows that for any $i < n$ with $\mathcal{I}_i \leq_\rho \mathcal{I}_{i+1}$ either $|\Delta^{\mathcal{I}_i}| < |\Delta^{\mathcal{I}_{i+1}}|$ or $|A^{\mathcal{I}_i}| < |A^{\mathcal{I}_{i+1}}|$ for some concept name A. By Lemma 58 we have $|\Delta^{\mathcal{I}_i}| \leq |\mathcal{T}|$ for all $i \leq n$. Hence $k - j \leq |\mathcal{T}|^2$ for any subsequence $\mathcal{I}_j \ldots, \mathcal{I}_k$ of $\mathcal{I}_0, \ldots, \mathcal{I}_n$ with $\mathcal{I}_i \leq_\rho \mathcal{I}_{i+1}$ for all $j \leq i < k$. It follows that $n \leq |\mathcal{T}| + |\mathcal{T}|^2$. ❏

We have thus established the main result of this section. Note that we obtain a polynomial time learning algorithm since checking $\mathcal{T} \models \alpha$ is in polynomial times for $\mathcal{EL}$ TBoxes $\mathcal{T}$ and $\mathcal{EL}$ CIs α (as discussed in Section 2).

Theorem 60 *$\mathcal{EL}_{\mathsf{lhs}}$ TBoxes are polynomial time learnable using membership and equivalence queries.*

The following example shows that Algorithm 6 does not terminate in polynomial time if in Line 5 it does not transform the given counterexample into an essential $\mathcal{T}$-countermodel.

Example 61 Assume that Line 5 of Algorithm 6 does not modify the counterexample $C \sqsubseteq A$ given in Line 4 if the second condition for essential $\mathcal{T}$-countermodels ($\mathcal{I}_C|^-_{d\downarrow} \models \mathcal{T}$ for all $d \in \Delta^{\mathcal{I}_C} \setminus \{\rho_{\mathcal{I}_C}\}$) is satisfied but the first condition ($\mathcal{I}_C|^-_{\rho} \models \mathcal{T}$) does not hold. Then for the target TBox $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$ the oracle can return the infinite sequence of positive counterexamples $\exists r^n.A \sqsubseteq A$, with n a prime number. In fact, Algorithm 6 would simply construct the list $\mathfrak{I}$ of interpretations $\mathcal{I}_{\exists r^n.A}$, n a prime number, and would not terminate. To show this observe that Algorithm 6 would never replace a CI in the list $\mathfrak{I}$ by another CI since $\mathcal{I}_{\exists r^n.A} \times_\rho \mathcal{I}_{\exists r^{n+m}.A} = \mathcal{I}_{\exists r^n.\top}$ and $\mathcal{I}_{\exists r^n.\top} \models \mathcal{T}$.

Now assume that Line 5 of Algorithm 6 does not modify the counterexample $C \sqsubseteq A$ given in Line 4 if the first condition for essential $\mathcal{T}$-countermodels is satisfied but the second condition does not hold. Let $\mathcal{T}$ be a TBox containing $\exists r.A \sqsubseteq A$ and some CIs containing the concept names B_1 and B_2, say, for simplicity, $B_1 \sqsubseteq B_1$ and $B_2 \sqsubseteq B_2$. Let $\varphi^1 = \exists r.(B_1 \sqcap B_2)$ and $\varphi^{n+1} = \exists r.(\varphi^n \sqcap B_1 \sqcap B_2)$. Then the oracle can return n positive counterexamples $\exists r.A \sqcap C_i \sqsubseteq A$, where the tree T_{C_i} corresponding to C_i is the result of identifying the i-th node of the tree T_{φ^i} corresponding to φ_i with the root of the tree corresponding to $\exists r.(B_1 \sqcap \varphi^n) \sqcap \exists r.(B_2 \sqcap \varphi^n)$. Note that the product of $\mathcal{I}_{C_1}, \dots, \mathcal{I}_{C_n}$ is an interpretation with $O(2^n)$ elements. Then, at the n-th iteration, Algorithm 6 computes an interpretation of exponential size in n.

6. Limits of Polynomial Learnability

The main result of this section is that $\mathcal{EL}$ TBoxes are not polynomial query learnable using membership and equivalence queries. We also show that DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes are not polynomial query learnable using membership or equivalence queries alone. The latter result also holds for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes. In this case, however, it follows already from the fact that propositional Horn logic is not polynomial query learnable from entailments using membership or equivalence queries alone (Frazier and Pitt, 1993; Angluin et al., 1992; Angluin, 1987a).

We start by proving the non-polynomial query learnability result for $\mathcal{EL}$ TBoxes. On our way, we also prove non-polynomial query learnability of DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes using membership queries only. Our proof shows that even acyclic $\mathcal{EL}$ TBoxes are not polynomial query learnable and, in fact, heavily relies on the additional properties of acyclic TBoxes. Recall that an $\mathcal{EL}$ TBox is called *acyclic* if it satisfies the following conditions (Baader et al., 2017; Konev et al., 2012):

- all CIs and CEs are of the form $A \sqsubseteq C$ or $A \equiv C$, where A is a concept name;
- no concept name occurs more than once on the left-hand side of a CI;
- there are no cyclic definitions: there is no sequence $\alpha_0, \dots, \alpha_n$ of CIs such that the concept name on the left-hand side of α_0 occurs in α_n and the concept name on the left-hand side of α_{i+1} occurs in the right-hand side of α_i for all $i < n$.

Our non-polynomial query learnability proof is inspired by Angluin's lower bound for the following abstract learning problem (Angluin, 1987b): a learner aims to identify one of N distinct sets $L_1, \dots, L_N$ which have the property that there exists a set $L_\cap$ for which $L_i \cap L_j = L_\cap$, for any $i \neq j$. It is assumed that $L_\cap$ is not a valid argument to an equivalence query. The learner can pose membership queries "$x \in L$?" and equivalence queries "$H = L$?". Then in the worst case it takes at least $N-1$ membership and equivalence queries to exactly identify a hypothesis L_i from $L_1, \dots, L_N$. The proof proceeds as follows. At every stage of computation, the oracle (which here should be viewed as an adversary) maintains a set of hypotheses S, which the learner is not able to distinguish based on the answers given so far. Initially, $S = \{L_1, \dots, L_N\}$. When the learner asks a membership query x, the oracle returns 'Yes' if $x \in L_\cap$ and 'No' otherwise. In the latter case, the (unique) L_i such that $x \in L_i$ is removed from S. When the learner asks an equivalence query H, the oracle returns 'No' and a counterexample $x \in L_\cap \oplus H$ (the symmetric difference of $L_\cap$ and H). This always exists as $L_\cap$ is not a valid query. If the counterexample x is not a member of $L_\cap$, (at most one) $L_i \in S$ such that $x \in L_i$ is eliminated from S. In the worst case, the learner has to reduce the cardinality of S to one to exactly identify a hypothesis, which takes $N-1$ queries.

Similarly to the method outlined above, in our proof we maintain a set of acyclic $\mathcal{EL}$ TBoxes whose members the learning algorithm is not able to distinguish based on the answers obtained so far. For didactic purposes, we first present a set of acyclic TBoxes $S_N = \{\mathcal{T}_1, \dots, \mathcal{T}_N\}$, where N is superpolynomial in the size of every TBox $\mathcal{T}_i$, for which the oracle can respond to membership queries in the way described above but which is polynomial time learnable when equivalence queries are also allowed. We then show how the TBoxes can be modified to obtain a family of acyclic TBoxes that is not polynomial query learnable using membership and equivalence queries.

To present the TBoxes in S_N, fix two role names r and s. We use the following abbreviation. For any sequence $\sigma = \sigma^1\sigma^2 \dots \sigma^n \in \{r, s\}^n$, the expression $\exists \boldsymbol{\sigma}.C$ stands for $\exists \sigma^1.\exists \sigma^2 \dots \exists \sigma^n.C$. Then for every such sequence $\boldsymbol{\sigma}$, of which there are $N = 2^n$ many, consider the acyclic $\mathcal{EL}$ TBox $\mathcal{T}_\sigma$ defined as

$$\begin{aligned} \mathcal{T}_{\boldsymbol{\sigma}} &= \{A \sqsubseteq \exists \boldsymbol{\sigma}.M \sqcap X_0\} \cup \mathcal{T}_0 \text{ with} \\ \mathcal{T}_0 &= \{X_i \sqsubseteq \exists r.X_{i+1} \sqcap \exists s.X_{i+1} \mid 0 \leq i < n\}. \end{aligned}$$

Observe that the canonical model $\mathcal{I}_{X_0,\mathcal{T}_0}$ of X_0 and $\mathcal{T}_0$ consists of a full binary tree whose edges are labelled with the role names r and s and with X_0 at the root ρ_{X_0}, X_1 at level 1, and so on. In the canonical model $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}}}$ of A and $\mathcal{T}_{\boldsymbol{\sigma}}$, the root is labelled by A and X_0 and, *in addition to* the binary tree, there is a path given by the sequence $\boldsymbol{\sigma}$ whose endpoint is marked by the concept name M.

One can use Angluin's strategy to show that TBoxes from the set S_N of all such TBoxes $\mathcal{T}_{\boldsymbol{\sigma}}$ cannot be learned using polynomially many polynomial size membership queries only: notice that for no sequence $\boldsymbol{\sigma'} \neq \boldsymbol{\sigma}$ of length n, we have $\mathcal{T}_{\boldsymbol{\sigma}} \models A \sqsubseteq \exists \boldsymbol{\sigma'}.M$. Thus a membership query of the form $A \sqsubseteq \exists \boldsymbol{\sigma}.M$ eliminates at most one TBox from the set of TBoxes that the learner cannot distinguish. This observation can be generalised to arbitrary membership queries $C \sqsubseteq D$ in $\mathcal{EL}$; however, we instead observe that the TBoxes $\mathcal{T}_{\boldsymbol{\sigma}}$ are formulated in DL-Lite$_{\mathcal{R}}^{\exists}$ and prove a stronger result. The proof, given in the appendix, uses the canonical model construction introduced in Section 2.

Lemma 62 *For every DL-Lite$_{\mathcal{R}}^{\exists}$ CI $B \sqsubseteq D$ over the signature of $\mathcal{T}_{\boldsymbol{\sigma}}$,*

- *either $\mathcal{T}_{\boldsymbol{\sigma}} \models B \sqsubseteq D$ for every $\mathcal{T}_{\boldsymbol{\sigma}} \in S_N$*
- *or there is at most one $\mathcal{T}_{\boldsymbol{\sigma}} \in S_N$ such that $\mathcal{T}_{\boldsymbol{\sigma}} \models B \sqsubseteq D$.*

The argument outlined above immediately gives us the following side result.

Theorem 63 *DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes (even without inverse roles) are not polynomial query learnable using only membership queries.*

We return now to our proof that $\mathcal{EL}$ TBoxes are not polynomial query learnable using both membership and equivalence queries. Notice that the set of TBoxes S_N is not suitable as a single equivalence query is sufficient to learn any TBox from S_N in two steps: given the equivalence query $\{A \sqsubseteq X_0\} \cup \mathcal{T}_0$, the oracle has no other option but to reveal the target TBox $\mathcal{T}_{\boldsymbol{\sigma}}$ as $A \sqsubseteq \exists \boldsymbol{\sigma}.M$ can be found 'inside' every counterexample.

Our strategy to rule out equivalence queries with the 'intersection TBox' is to modify $\mathcal{T}_1, \dots, \mathcal{T}_N$ in such a way that although a TBox $\mathcal{T}_\cap$ axiomatising the intersection over the set of consequences of each $\mathcal{T}_i$, $i \leq N$, exists, its size is superpolynomial and so it cannot be used as an equivalence query by a polynomial query learning algorithm.

For every $n > 0$ and every n-tuple $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, where every $\boldsymbol{\sigma}_i$ is a role sequence of length n as above, we define an acyclic $\mathcal{EL}$ TBox $\mathcal{T}_L$ as the union of $\mathcal{T}_0$ and the following CIs and CEs:[5]

$$\begin{array}{ccc} A_1 \sqsubseteq \exists \boldsymbol{\sigma}_1.M \sqcap X_0 & & A_n \sqsubseteq \exists \boldsymbol{\sigma}_n.M \sqcap X_0 \\ B_1 \sqsubseteq \exists \boldsymbol{\sigma}_1.M \sqcap X_0 & \cdots & B_n \sqsubseteq \exists \boldsymbol{\sigma}_n.M \sqcap X_0 \end{array}$$

$$A \equiv X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \dots \sqcap \exists \boldsymbol{\sigma}_n.M.$$

Observe that every $\mathcal{T}_L$ contains the TBoxes $\mathcal{T}_{\boldsymbol{\sigma}_i}$, $1 \leq i \leq n$, discussed above with A replaced by any of the three concept names A, A_i, B_i. In addition, *every* $\mathcal{T}_L$ entails, among other CIs, $\bigsqcap_{i=1}^n C_i \sqsubseteq A$, where every C_i is either A_i or B_i. There are 2^n different such CIs, which indicates that every representation of the 'intersection TBox' requires superpolynomially many axioms. It follows from Lemma 67 below that this is indeed the case.

Let $\mathfrak{L}_n$ be a set of n-tuples such that for $1 \leq i \leq n$ and every $L, L' \in \mathfrak{L}_n$ with $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, $L' = (\boldsymbol{\sigma}'_1, \dots, \boldsymbol{\sigma}'_n)$, if $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}'_j$ then $L = L'$ and $i = j$. Then for any sequence $\boldsymbol{\sigma}$ of length n there exists at most one $L \in \mathfrak{L}_n$ and at most one $i \leq n$ such that $\mathcal{T}_L \models A_i \sqsubseteq \exists \boldsymbol{\sigma}.M$ and $\mathcal{T}_L \models B_i \sqsubseteq \exists \boldsymbol{\sigma}.M$. We can choose $\mathfrak{L}_n$ such that there are $N = \lfloor 2^n/n \rfloor$ different tuples in $\mathfrak{L}_n$. Notice that the size of each $\mathcal{T}_L$ with $L \in \mathfrak{L}_n$ is polynomial in n and so N is superpolynomial in the size of each $\mathcal{T}_L$ with $L \in \mathfrak{L}_n$. Let the set of TBoxes that the learner cannot distinguish initially be $S_{\mathfrak{L}} = \{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$. We use Σ_n to denote the signature of $\mathcal{T}_L$.

For the proof of non-polynomial query learnability, we show that the oracle has a strategy to answer both membership and equivalence queries without eliminating too many TBoxes from $S_{\mathfrak{L}}$. We start with the former.

5. In fact, to prove non-polynomial query learnability, it suffices to consider $\exists \boldsymbol{\sigma}_1.M \sqcap \dots \sqcap \exists \boldsymbol{\sigma}_n.M \sqsubseteq A$ in place of the concept equivalence; however, CIs of this form are not allowed in acyclic TBoxes. CIs with a complex left-hand side or concept equivalences are essential for non-polynomial query learnability as any acyclic TBox containing expressions of the form $A \sqsubseteq C$ only is a DL-Lite$_{\mathcal{R}}^{\exists}$ TBox and thus polynomially learnable with membership and equivalence queries (Section 3).

Unlike the DL-Lite$^{\exists}_{\mathcal{R}}$ case presented above, membership query can eliminate more than one TBox from $S_{\mathfrak{L}}$. Consider, for example, two TBoxes $\mathcal{T}_L$ and $\mathcal{T}_{L'}$, where $\{L, L'\} \subseteq \mathfrak{L}_n$ with $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n)$ and $L' = (\boldsymbol{\sigma}'_1, \ldots, \boldsymbol{\sigma}'_n)$. Then the CI

$$X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \exists \boldsymbol{\sigma}'_1.M \sqcap A_2 \sqcap \cdots \sqcap A_n \sqsubseteq A$$

is entailed by both $\mathcal{T}_L$ and $\mathcal{T}_{L'}$ but not by any other $\mathcal{T}_{L''}$ with $L'' \in \mathfrak{L}_n$. We prove, however, that the number of TBoxes eliminated from $S_{\mathfrak{L}}$ by a single membership query can be linearly bounded by the size of the query.

Lemma 64 *For all $\mathcal{EL}$ CIs $C \sqsubseteq D$ over Σ_n:*

- *either $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$*
- *or the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq D$ does not exceed $|C|$.*

The proof of Lemma 64 is technical and is deferred to the appendix. To illustrate our proof method here we consider a particular case that deals with membership queries of the form $C \sqsubseteq \exists \boldsymbol{\sigma}.M$ and is used in the proof of the general case. Both proofs rely on the following lemma from the study of the logical difference between ontologies (Konev et al., 2012). It characterises CIs entailed by acyclic $\mathcal{EL}$ TBoxes.

Lemma 65 (Konev et al. (2012)) *Let $\mathcal{T}$ be an acyclic $\mathcal{EL}$ TBox, r a role name and D an $\mathcal{EL}$ concept expression. Suppose that $\mathcal{T} \models \bigsqcap_{1 \leq i \leq n} A_i \sqcap \bigsqcap_{1 \leq j \leq m} \exists r_j.C_j \sqsubseteq D$, where A_i are concept names for $1 \leq i \leq n$, C_j are $\mathcal{EL}$ concept expressions for $1 \leq j \leq m$, and $m, n \geq 0$. Then the following holds:*

- *if D is a concept name such that $\mathcal{T}$ does not contain any CE $D \equiv C$ for any concept expression C, then there exists A_i, $1 \leq i \leq n$, such that $\mathcal{T} \models A_i \sqsubseteq D$;*
- *if D is of the form $\exists r.D'$ then either (i) there exists A_i, $1 \leq i \leq n$, such that $\mathcal{T} \models A_i \sqsubseteq \exists r.D'$ or (ii) there exists r_j, $1 \leq j \leq m$, such that $r_j = r$ and $\mathcal{T} \models C_j \sqsubseteq D'$.*

The following lemma considers membership queries of the form $C \sqsubseteq \exists \boldsymbol{\sigma}.M$.

Lemma 66 *For any $0 \leq m \leq n$, any sequence of role names $\boldsymbol{\sigma} = \sigma^1 \ldots \sigma^m \in \{r, s\}^m$, and any $\mathcal{EL}$ concept expression C over Σ_n:*

- *either $\mathcal{T}_L \models C \sqsubseteq \exists \boldsymbol{\sigma}.M$ for every $\mathcal{T}_L$ with $L \in \mathfrak{L}_n$;*
- *or there is at most one $\mathcal{T}_L$ such that $\mathcal{T}_L \models C \sqsubseteq \exists \boldsymbol{\sigma}.M$.*

Proof The lemma follows from the following claim.

Claim. Let $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n) \in \mathfrak{L}_n$ be such that $\mathcal{T}_L \models C \sqsubseteq \exists \boldsymbol{\sigma}.M$. Then *either* (1) there exists $i \leq n$ such that $\boldsymbol{\sigma} = \boldsymbol{\sigma}_i$ and C is of the form $A \sqcap C'$, $A_i \sqcap C'$ or $B_i \sqcap C'$, for some $\mathcal{EL}$ concept expression C'; *or* (2) we have $\emptyset \models C \sqsubseteq \exists \boldsymbol{\sigma}.M$.

Proof of Claim. We prove the claim by induction on m. If $m = 0$, by Lemma 65, the concept expression C is of the form $Z \sqcap C'$, for some concept name Z and concept expression C' such that $\mathcal{T}_L \models Z \sqsubseteq M$. As $\mathcal{T}_L \models Z \sqsubseteq M$ does not hold for any concept name Z distinct from M, we obtain $Z = M$. Thus, $\emptyset \models C \sqsubseteq M$ and Point (2) follows.
Let $m > 0$. By Lemma 65 we have one of the following two cases:

- C is of the form $X \sqcap C'$, for some concept name X and concept expression C' such that $\mathcal{T}_L \models X \sqsubseteq \exists \boldsymbol{\sigma}.M$. But then there exists $i \leq n$ such that $\boldsymbol{\sigma} = \boldsymbol{\sigma}_i$ and $X \in \{A, A_i, B_i\}$ and Point (1) follows.

- C is of the form $\exists \sigma^1.C' \sqcap C''$, for some concept expressions C' and C'', and $\mathcal{T}_L \models C' \sqsubseteq \exists \sigma^2. \cdots \exists \sigma^m.M$. Notice that the length of the sequence $\sigma^2 \dots \sigma^n$ is strictly less than n. Thus, by induction hypothesis, $\emptyset \models C' \sqsubseteq \exists \sigma^2. \cdots \exists \sigma^m.M$. But then $\emptyset \models C \sqsubseteq \exists \boldsymbol{\sigma}.M$ and Point (2) follows.

This finishes the proof of the claim. To see that the claim entails the lemma observe that at most one $L \in \mathfrak{L}_n$ can satisfy Point (1). Point (2) entails that $\mathcal{T}_L \models C \sqsubseteq \exists \boldsymbol{\sigma}.M$ for every $\mathcal{T}_L$ with $L \in \mathfrak{L}_n$. ❏

We now show how the oracle can answer equivalence queries, aiming to show that for any polynomial size equivalence query $\mathcal{H}$, the oracle can return a counterexample $C \sqsubseteq D$ such that either (i) $\mathcal{H} \models C \sqsubseteq D$ and $\mathcal{T}_L \models C \sqsubseteq D$ for at most one $L \in \mathfrak{L}_n$ or (ii) $\mathcal{H} \not\models C \sqsubseteq D$ and $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$. Thus, such a counterexample eliminates at most one $\mathcal{T}_L$ from the set $S_{\mathfrak{L}}$ of TBoxes that the learner cannot distinguish. In addition, however, we have to take extra care of the size of counterexamples as the learning algorithm is allowed to formulate queries polynomial not only in the size of the target TBox but also in the size of the counterexamples returned by the oracle. For instance, if the hypothesis TBox $\mathcal{H}$ contains a CI $C \sqsubseteq D$ which is not entailed by any $\mathcal{T}_L$, one cannot simply return $C \sqsubseteq D$ as a counterexample since the learner will be able to 'pump up' its capacity by asking a sequence of equivalence queries $\mathcal{H}_i = \{C_i \sqsubseteq D_i\}$ such that the size of $C_{i+1} \sqsubseteq D_{i+1}$ is twice the size of $C_i \sqsubseteq D_i$. Then at every stage in a run of the learning algorithm, the query size will be polynomial in the size of the input and the size of the largest counterexample received so far, but exponential size queries will become available to the learner. The following lemma addresses this issue.

Lemma 67 *For any $n > 1$ and any $\mathcal{EL}$ TBox $\mathcal{H}$ in Σ_n with $|\mathcal{H}| < 2^n$, there exists an $\mathcal{EL}$ CI $C \sqsubseteq D$ over Σ_n such that the size of $C \sqsubseteq D$ does not exceed $6n$ and*

- *if $\mathcal{H} \models C \sqsubseteq D$, then $\mathcal{T}_L \models C \sqsubseteq D$ for at most one $L \in \mathfrak{L}_n$;*

- *if $\mathcal{H} \not\models C \sqsubseteq D$, then $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$.*

Proof We define an exponentially large TBox $\mathcal{T}_\cap$ and use it to prove that one can select the required $\mathcal{EL}$ CI $C \sqsubseteq D$ in such a way that either $\mathcal{H} \models C \sqsubseteq D$ and $\mathcal{T}_\cap \not\models C \sqsubseteq D$, or vice versa.

To define $\mathcal{T}_\cap$, denote for any sequence $\boldsymbol{b} = b_1 \dots b_n \in \{0,1\}^n$ by $C_{\boldsymbol{b}}$ the conjunction $\bigsqcap_{i \leq n} C_i$, where $C_i = A_i$ if $b_i = 1$ and $C_i = B_i$ if $b_i = 0$. Then we define

$$\mathcal{T}_\cap = \mathcal{T}_0 \cup \{C_{\boldsymbol{b}} \sqsubseteq A \sqcap X_0 \mid \boldsymbol{b} \in \{0,1\}^n\}.$$

Consider the following cases for $\mathcal{H}$ and $\mathcal{T}_\cap$.

1. Suppose $\mathcal{H} \not\models \mathcal{T}_\cap$. Then there exists a CI $C \sqsubseteq D \in \mathcal{T}_\cap$ such that $\mathcal{H} \not\models C \sqsubseteq D$. Clearly, $C \sqsubseteq D$ is entailed by every $\mathcal{T}_L$, for $L \in \mathfrak{L}_n$, and the size of $C \sqsubseteq D$ does not exceed $6n$. Thus $C \sqsubseteq D$ is as required.

2. Suppose there exist $\boldsymbol{b} \in \{0,1\}^n$ and a concept expression of the form $\exists t.D'$ such that $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq \exists t.D'$ and $\mathcal{T}_0 \not\models C_{\boldsymbol{b}} \sqsubseteq \exists t.D'$. It can be seen (Lemma 73 in the appendix), that there exists a sequence of role names $t_1, \dots, t_l \in \{r,s\}^l$ with $0 \leq l \leq n+1$ and $Y \in \{\top\} \cup \mathsf{N_C}$ such that $\emptyset \models \exists t.D' \sqsubseteq \exists t_1. \cdots \exists t_l.Y$. Thus, $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ and $\mathcal{T}_0 \not\models X_0 \sqsubseteq \exists t_1. \cdots \exists t_l.Y$. We show that the inclusion $C_{\boldsymbol{b}} \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ is as required. Clearly, the size of $C_{\boldsymbol{b}} \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ does not exceed $6n$. It remains to prove that $\mathcal{T}_L \models C_{\boldsymbol{b}} \sqsubseteq \exists t_1 \cdots \exists t_l.Y$ for at most one $L \in \mathfrak{L}_n$.

   Suppose there exists $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C_{\boldsymbol{b}} \sqsubseteq \exists t_1. \cdots \exists t_l.Y$. By Lemma 65, there exists A_j or B_j such that $\mathcal{T}_L \models A_j \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ or $\mathcal{T}_L \models B_j \sqsubseteq \exists t_1. \cdots \exists t_l.Y$, respectively. As $\mathcal{T}_0 \not\models X_0 \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ it is easy to see that $l = n$, $t_1 t_2 \dots t_n = \boldsymbol{\sigma}_j$, and $Y = M$ follow. As $\mathcal{T}_{L'} \not\models C_{\boldsymbol{b}} \sqsubseteq \exists \boldsymbol{\sigma}_j.M$ for any $L' \in \mathfrak{L}_n$ such that $L' \neq L$, it follows that $\mathcal{T}_L \models C_{\boldsymbol{b}} \sqsubseteq \exists t_1 \cdots \exists t_l.Y$ for at most one $L \in \mathfrak{L}_n$

3. Finally, suppose that neither Case 1 nor 2 above apply. Then $\mathcal{H} \models \mathcal{T}_\cap$ and for every $\boldsymbol{b} \in \{0,1\}^n$ and every $\mathcal{EL}$ concept expression over Σ_n of the form $\exists t.D'$: if $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq \exists t.D'$ then $\mathcal{T}_0 \models X_0 \sqsubseteq \exists t.D'$. We show that unless there exists a CI $C \sqsubseteq D$ satisfying the conditions of the lemma, $\mathcal{H}$ contains at least 2^n different CIs (and thus derive a contradiction).

   Fix some $\boldsymbol{b} = b_1 \dots b_n \in \{0,1\}^n$. From $\mathcal{H} \models \mathcal{T}_\cap$ we obtain $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq A$. Then there must exist at least one CI $C \sqsubseteq A \sqcap D \in \mathcal{H}$ such that $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq C$ and $\emptyset \not\models C \sqsubseteq A$. Let $C = Z_1 \sqcap \dots \sqcap Z_m \sqcap \exists t_1.C'_1 \sqcap \dots \sqcap \exists t_l.C'_l$, where $Z_1, \dots, Z_m$ are different concept names. As $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq \exists t_j.C'_j$ we have $\mathcal{T}_0 \models X_0 \sqsubseteq \exists t_j.C'_j$, for $j = 1, \dots l$. As $\mathcal{H} \models \mathcal{T}_\cap$ we have $\mathcal{H} \models X_0 \sqsubseteq \exists t_j.C'_j$, for $j = 1, \dots l$. So $\mathcal{H} \models Z_1 \sqcap \dots \sqcap Z_m \sqcap X_0 \sqsubseteq A$.

   - Suppose there exists i such that there is no $Z_j \in \{A_i, B_i\}$. Then we have $\mathcal{T}_L \not\models Z_1 \sqcap \dots \sqcap Z_m \sqcap X_0 \sqsubseteq A$, for any $L \in \mathfrak{L}_n$. Notice that $Z_1 \sqcap \dots \sqcap Z_m$ contains at most all concepts names in Σ_n, except A_i, B_i. Thus, the size of $Z_1 \sqcap \dots \sqcap Z_m \sqcap X_0 \sqsubseteq A$ does not exceed $6n$, and $Z_1 \sqcap \dots \sqcap Z_m \sqcap X_0 \sqsubseteq A$ is as required.
   - Assume that $Z_0 \sqcap \dots \sqcap Z_m \sqcap X_0$ contains a conjunct B_i such that $b_i \neq 0$. Then $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq B_i$ and there is no $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C_{\boldsymbol{b}} \sqsubseteq B_i$. The size of $C_{\boldsymbol{b}} \sqsubseteq B_i$ does not exceed $6n$, so $C_{\boldsymbol{b}} \sqsubseteq B_i$ is as required.

- Assume that $Z_0 \sqcap \cdots \sqcap Z_m \sqcap X_0$ contains a conjunct A_i such that $b_i \neq 1$. Then $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq A_i$ and there is no $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C_{\boldsymbol{b}} \sqsubseteq A_i$. The size of $C_{\boldsymbol{b}} \sqsubseteq A_i$ does not exceed $6n$, so $C_{\boldsymbol{b}} \sqsubseteq A_i$ is as required.
- If none of the above applies, then $Z_1 \sqcap \cdots \sqcap Z_m \sqcap X_0$ contains exactly the A_i with $b_i = 1$ and exactly the B_i with $b_i = 0$.

This argument applies to arbitrary $\boldsymbol{b} \in \{0,1\}^n$. Thus, if there exists no CI $C \sqsubseteq D$ satisfying the conditions of the lemma then, by the final case, $\mathcal{H}$ contains at least 2^n CIs.

❑

Now we have all the ingredients to prove that $\mathcal{EL}$ TBoxes are not polynomial query learnable using membership and equivalence queries.

Theorem 68 *$\mathcal{EL}$ TBoxes are not polynomial query learnable using membership and equivalence queries.*

Proof Assume that TBoxes are polynomial query learnable. Then there exists a learning algorithm whose query complexity (the sum of the sizes of the inputs to membership and equivalence queries made by the algorithm up to a computation step) is bounded at any stage by a polynomial $p(n, m)$. Choose n such that $\lfloor 2^n/n \rfloor > (p(n, 6n))^2$ and let $S_{\mathfrak{L}} = \{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$. We follow Angluin's strategy of letting the oracle remove TBoxes from $S_{\mathfrak{L}}$ in such a way that the learner cannot distinguish between any of the remaining TBoxes. Given a membership query $C \sqsubseteq D$, if $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$, then the answer is 'yes'; otherwise the answer is 'no' and all $\mathcal{T}_L$ with $\mathcal{T}_L \models C \sqsubseteq D$ are removed from $S_{\mathfrak{L}}$ (by Lemma 64, there are at most $|C|$ such TBoxes). Given an equivalence query $\mathcal{H}$, the answer is 'no', a counterexample $C \sqsubseteq D$ guaranteed by Lemma 67 is produced, and (at most one) $\mathcal{T}_L$ such that $\mathcal{T}_L \models C \sqsubseteq D$ is removed from $S_{\mathfrak{L}}$.

As all counterexamples produced are smaller than $6n$, the overall query complexity of the algorithm is bounded by $p(n, 6n)$. Hence, the learner asks no more than $p(n, 6n)$ queries and the size of every query does not exceed $p(n, 6n)$. By Lemmas 64 and 67, at most $(p(n, 6n))^2$ TBoxes are removed from $S_{\mathfrak{L}}$ during the run of the algorithm. But then, the algorithm cannot distinguish between any remaining TBoxes and we have derived a contradiction. ❑

We conclude this section by showing that DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes cannot be learned using polynomially many polynomial size equivalence queries only. We use the following result on non-polynomial query learnability of monotone DNF formulas, that is, DNF formulas that do not use negation, using equivalence queries due to Angluin (1990). Here, equivalence queries take a hypothesis ψ in the form of a monotone DNF formula and return as a counterexample either a truth assignment that satisfies ψ but not the target formula ϕ or vice versa. Let $M(n, t, s)$ denote the set of all monotone DNF formulas whose variables are $x_1, \ldots, x_n$, that have exactly t conjunctions, and where each conjunction contains exactly s variables.

Theorem 69 (Angluin (1990)) *For any polynomial $q(\cdot)$ there exist constants t_0 and s_0 and a strategy [6] for the oracle $\mathfrak{O}$ to answer equivalence queries posed by a learning algorithm*

6. The existence of this strategy is a direct consequence of Theorem 8 by Angluin (1990), which states that the class of DNF formulae has the *approximate fingerprint* property, and the proof of Theorem 1

in such a way that for sufficiently large n any learning algorithm that asks at most $q(n)$ equivalence queries, each bounded in size by $q(n)$, cannot exactly identify elements of $M(n, t_0, s_0)$.

To employ Theorem 69, we associate with every monotone DNF formula

$$\phi = \bigvee_{i=1}^{t} (x_1^i \wedge \cdots \wedge x_{s^i}^i),$$

where $\{x_1^i, \ldots, x_{s^i}^i\} \subseteq \{x_1, \ldots, x_n\}$, a DL-Lite$_{\mathcal{R}}^{\exists}$ TBox $\mathcal{T}_\phi$ as follows. With each conjunct $x_1^i \wedge \cdots \wedge x_{s^i}^i$ we associate a concept expression

$$C_i := \exists \rho_1^i . \exists \rho_2^i . \ldots \exists \rho_n^i . \top,$$

where $\rho_j^i = r$ if x_j occurs in $x_1^i \wedge \cdots \wedge x_{s^i}^i$ and $\rho_j^i = \bar{r}$ otherwise (r and $\bar{r}$ are role names). Let A be a concept name and set

$$\mathcal{T}_\phi = \{A \sqsubseteq \textstyle\bigsqcap_{i=1}^{t} C_i, \quad \bar{r} \sqsubseteq r\}.$$

For example, for $n = 4$ and $\phi = (x_1 \wedge x_4) \vee x_2$ we have

$$\mathcal{T}_\phi = \{A \sqsubseteq \exists r . \exists \bar{r} . \exists \bar{r} . \exists r . \top,\ A \sqsubseteq \exists \bar{r} . \exists r . \exists \bar{r} . \exists \bar{r} . \top,\ \bar{r} \sqsubseteq r\}.$$

We say that a TBox $\mathcal{T}$ *has a DNF-representation for n* if it is obtained by the translation of a monotone DNF-formula with n variables; that is, if $\mathcal{T}$ is of the following form, for some $\Gamma \subseteq \{r, \bar{r}\}^n$:

$$\{A \sqsubseteq \bigsqcap_{\rho_1 \cdots \rho_n \in \Gamma} \exists \rho_1 . \exists \rho_2 . \ldots \exists \rho_n . \top, \quad \bar{r} \sqsubseteq r\}.$$

A truth assignment I (for the variables $x_1 \ldots, x_n$) also corresponds to a concept expression

$$C_I := \exists \rho_1^i . \exists \rho_2^i . \ldots \exists \rho_n^i . \top,$$

where $\rho_j^i = r$ if I makes x_j true and $\rho_j^i = \bar{r}$ otherwise. Then

$$I \models \phi \text{ if, and only if, } \mathcal{T}_\phi \models A \sqsubseteq C_I$$

holds for all truth assignments I.

Note that $\bar{r}$ represents that a variable is false and r that a variable is true. Thus, the RI $\bar{r} \sqsubseteq r$ captures the monotonicity of the DNF formulas considered. For any fixed values n, s and t, we set

$$T(n, t, s) = \{\mathcal{T}_\phi \mid \phi \in M(n, t, s)\}.$$

Note that the TBoxes in $T(n, t, s)$ are exactly those TBoxes that have a DNF-representation for n and satisfy additionally the conditions that the DNF represented by $\mathcal{T}_\phi$ has exactly t conjunctions each conjunction of which has exactly s variables.

by Angluin (1990), where such a strategy is explicitly constructed for any class having approximate fingerprints.

We describe now the strategy for the oracle $\mathfrak{O}'$ to answer equivalence queries so that no learning algorithm is able to exactly identify members of $T(n,t,s)$ based on the answers to polynomially many equivalence queries of polynomial size. If the TBox in the equivalence query is 'obviously' not within the class $T(n,t,s)$, then we will explicitly produce a counterexample that the oracle can return. If, on the other hand, the TBox $\mathcal{H}$ from the equivalence query is 'similar' to TBoxes that have a DNF-representation for n, then we approximate $\mathcal{H}$ by a TBox $\mathcal{H}'$ that has a DNF-representation for n and return the counterexample $A \sqsubseteq C_I$ corresponding to the truth assignment I that the oracle $\mathfrak{O}$ from Theorem 69 would return when given ψ.

In detail the strategy is as follows. Assume q is the given polynomial in Theorem 69 and that t_0, s_0 and the strategy of the oracle $\mathfrak{O}$ are chosen so that for sufficiently large n no learning algorithm for DNF formulas that asks at most $q(n)$ equivalence queries, each bounded in size by $q(n)$, can distinguish all members of $M(n,t_0,s_0)$. Choose a sufficiently large n. Let $\mathcal{H}$ be an equivalence TBox query issued by a learning algorithm. Then $\mathfrak{O}'$ does the following:

1. If $\mathcal{H}$ entails some $A \sqsubseteq \exists\rho_1.\exists\rho_2.\ldots\exists\rho_{n+1}.\top$ with $\rho_i \in \{r,\bar{r}\}$ for $1 \leq i \leq n+1$, then return this CI as a negative counterexample;

2. If $\mathcal{H}$ entails some $\exists\rho_1.\top \sqsubseteq \exists\rho_2.\top$ such that $\{\rho_1,\rho_2\} \subseteq \{r,\bar{r},r^-,\bar{r}^-\}$ and $\{\bar{r} \sqsubseteq r\} \not\models \exists\rho_1.\top \sqsubseteq \exists\rho_2.\top$, then return this CI as a negative counterexample;

3. If $\mathcal{H} \models \exists\rho_1.\top \sqsubseteq \exists\rho_2.\exists\rho_3.\top$ such that $\{\rho_1,\rho_2,\rho_3\} \subseteq \{r,\bar{r}\}$, then return this CI as a negative counterexample;

4. If there exists no $\rho_1,\ldots,\rho_n \in \{r,\bar{r}\}^n$ such that $\mathcal{H} \models A \sqsubseteq \exists\rho_1.\cdots\exists\rho_n.\top$ then return $A \sqsubseteq \underbrace{\exists r\cdots\exists r}_{n}.\top$ as a positive counterexample.

5. Suppose now that none of the above applies. We say that a sequence $\rho_1,\ldots,\rho_n \in \{r,\bar{r}\}^n$ is *r-minimal for $\mathcal{H}$* if $\mathcal{H} \models A \sqsubseteq \exists\rho_1.\cdots\exists\rho_n.\top$ and whenever $\rho_i = r$, for $1 \leq i \leq n$, we have $\mathcal{H} \not\models \exists\rho_1.\cdots\exists\rho_{i-1}.\exists\bar{r}.\exists\rho_{i+1}.\cdots\exists\rho_n.\top$. We obtain a TBox $\mathcal{H}'$ with a DNF representation by setting

$$\mathcal{H}' = \{A \sqsubseteq \bigsqcap_{\substack{\rho_1,\ldots,\rho_n \text{ is} \\ r\text{-minimal for } \mathcal{H}}} \exists\rho_1.\cdots\exists\rho_n.\top, \quad \bar{r} \sqsubseteq r\}.$$

   Observe that for any sequence $\rho_1,\ldots,\rho_n \in \{r,\bar{r}\}^n$ we have $\mathcal{H} \models A \sqsubseteq \exists\rho_1.\cdots\exists\rho_n.\top$ if, and only if, $\mathcal{H}' \models A \sqsubseteq \exists\rho_1.\cdots\exists\rho_n.\top$. We convert $\mathcal{H}'$ into its corresponding monotone DNF formula $\phi_{\mathcal{H}'}$ by reversing the translation from monotone DNF formulas into DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes of the above form in the obvious way. Note that the size of $\phi_{\mathcal{H}'}$ is linear in the size of $\mathcal{H}'$. Given $\phi_{\mathcal{H}'}$ the oracle $\mathfrak{O}$ returns a (positive or negative) counterexample (a truth assignment) I. Then return the counterexample in the form of the CI $A \sqsubseteq C_I$.

Observe that the answers given in Points 1 to 3 are correct in the sense that if an inclusion α is returned as a negative example then $\mathcal{T} \not\models \alpha$ for any $\mathcal{T} \in T(n,t,s)$. Point 4 is trivially correct, since any monotone DNF is satisfied by the truth assignment that makes every variable true. We analyse the size of the TBox $\mathcal{H}'$ computed in Point 5.

Lemma 70 *Assume that Points 1 to 4 do not apply to $\mathcal{H}$. Then the number of sequences $\rho_1, \dots, \rho_n \in \{r, \bar{r}\}^n$ which are r-minimal for $\mathcal{H}$ is bounded by $|\mathcal{H}|$.*

Proof We first show that if $\rho_1, \dots, \rho_n \in \{r, \bar{r}\}^n$ is r-minimal for $\mathcal{H}$, then there exists a CI $A \sqsubseteq C \in \mathcal{H}$ such that

(∗) there are concept expressions $C_0, \dots, C_n$ with $C_0 = C$ and $\exists \rho_{i+1}.C_{i+1}$ a top-level conjunct of C_i, for all $i < n$.

For the proof we require the canonical model $\mathcal{I}_{A,\mathcal{H}}$ of A and $\mathcal{H}$ (Lemma 12). Denote the root of $\mathcal{I}_{A,\mathcal{H}}$ by ρ_A. Let $\rho_1, \dots, \rho_n \in \{r, \bar{r}\}^n$ be r-minimal for $\mathcal{H}$. Then there are $d_0, \dots, d_n \in \Delta^{\mathcal{I}_{A,\mathcal{H}}}$ with $d_0 = \rho_A$ such that $(d_i, d_{i+1}) \in \rho_i^{\mathcal{I}_{A,\mathcal{H}}}$ for all $i < n$. By the canonical model construction and the assumption that Points 2 and 3 do not hold, there either exists $d_i \in A^{\mathcal{I}_{A,\mathcal{H}}}$ or there is a CI $A \sqsubseteq C \in \mathcal{H}$ such that (∗) holds. We show that the first condition does not hold. Assume for a prove by contradiction that $d_i \in A^{\mathcal{I}_{A,\mathcal{H}}}$. By Lemma 12, $\mathcal{H} \models A \sqsubseteq \exists \rho_1 \cdots \exists \rho_i.A$. But then $\mathcal{H} \models A \sqsubseteq \exists (\rho_1 \cdots \rho_i)^n.\top$ for all $n > 0$ which contradicts the assumption that Point 1 does not apply to $\mathcal{H}$.

It follows that the number of distinct r-minimal sequences is bounded by the number of distinct sequences $C_0, \dots, C_n$ with $A \sqsubseteq C_0 \in \mathcal{H}$ and $\exists \rho_{i+1}.C_{i+1}$ a top-level conjunct of C_i for all $i < n$. Thus, the number of distinct r-minimal sequences is bounded by $|\mathcal{H}|$. ❑

It follows from Lemma 70 that the size of the TBox $\mathcal{H}'$ computed in Point 5 is bounded by $4n|\mathcal{H}| + 2$.

Theorem 71 *DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes (even without inverse roles) are not polynomial query learnable using only equivalence queries.*

Proof Suppose that the query complexity of a learning algorithm $\mathfrak{A}$ for DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes in $\Sigma = \{A, r, \bar{r}\}$ is bounded at every stage of computation by a polynomial $p(x, y)$, where x is the size of the target TBox, and y is the maximal size of a counterexample returned by the oracle up to the current stage of computation. Let $q(n) = (p(n^2, 4n + 6))^2$, and let constants t_0 and s_0 be as guaranteed by Lemma 69. We claim that, for sufficiently large n, $\mathfrak{A}$ cannot distinguish some $\mathcal{T}_\phi$ and $\mathcal{T}_\psi$ for $\phi, \psi \in M(n, t_0, s_0)$.

Assuming that $n > 11$ (the maximal size of counterexamples given under Point 2 and 3), the largest counterexample returned by our strategy described above is of the form $A \sqsubseteq \exists \rho_1. \cdots \exists \rho_{n+1}.\top$, so for sufficiently large n the maximal size of any counterexample in any run of $\mathfrak{A}$ is bounded by $4n + 6 = 4(n + 1) + 2$. Similarly, the size of every potential target TBox $\mathcal{T}_\phi \in T(n, t_0, s_0)$ does not exceed $t_0 \cdot (4n + 2)$ and, as t_0 is a constant, for sufficiently large n it is bounded by n^2. Thus, for sufficiently large n the total query complexity of $\mathfrak{A}$ on any input from $T(n, t_0, s_0)$ is bounded by $p(n^2, 4n + 6)$. Obviously, the size of each query is bounded by the query complexity of the learning algorithm. So, the size of a DNF equivalence query forwarded to the strategy $\mathfrak{O}$ guaranteed by Lemma 69 is bounded by $4n \times p(n^2, 4n + 6) + 2 \leq q(n)$, and there will be at most $q(n)$ queries forwarded. But then $\mathfrak{O}$ can return answers such that some ϕ and ψ from $M(n, t_0, s_0)$ cannot be distinguished. It remains to observe that $\mathfrak{A}$ cannot distinguish $\mathcal{T}_\phi$ and $\mathcal{T}_\psi$. ❑

7. Related Work

Some related work has already been discussed in the introduction to this paper. Here we discuss in more detail related work from *ontology learning* in general and *exact learning of ontologies* in particular. We start with the former.

Ontology Learning. Research in ontology learning has a rich history that we cannot discuss here in full detail. The collection edited by Lehmann and Völker (2014) and surveys authored by Cimiano et al. (2010) and Wong et al. (2012) provide an excellent introduction to the state of the art in this field. The techniques applied in ontology learning range from information extraction and text mining to interactive learning and inductive logic programming (ILP). Of particular relevance for this paper are the approaches to learning logical expressions (rather than subsumption hierarchies between concept names). For example, the work of Lehmann and Haase (2009), Lehmann and Hitzler (2010), and Bühmann et al. (2014) applies techniques from ILP to learn description logic concept expressions. ILP is applied as well by Lisi (2011) for learning logical rules for ontologies. The learning of fuzzy DLs has been considered by Lisi and Straccia (2015). Other machine learning methods which have been applied to learn ontology axioms include Association Rule Mining (ARM) (Völker and Niepert, 2011; Fleischhacker et al., 2012; Völker et al., 2015) and Formal Concept Analysis (FCA) (Rudolph, 2004; Baader et al., 2007; Distel, 2011; Borchmann, 2014; Ganter et al., 2016). Recently, learnability of lightweight DL TBoxes from finite sets of interpretations has been investigated (Klarman and Britz, 2015).

Exact Learning of Description Logic Concept Expressions. Rather than aiming to learn a TBox here one is interested in learning a target concept expression C_*. This was first studied by Cohen and Hirsh (1994a,b) and Frazier and Pitt (1996). The standard learning protocol is as follows:

- a membership query asks whether a concept expression C is subsumed by the target concept expression C_* (in symbols, $\emptyset \models C \sqsubseteq C_*?$);
- an equivalence query asks whether a concept expression C is equivalent to the target concept expression C_* (in symbols, $\emptyset \models C \equiv C_*?$). If C and C_* are not equivalent then the oracle gives a counterexample, that is, a concept expression C' such that either $\emptyset \models C' \sqsubseteq C_*$ and $\emptyset \not\models C' \sqsubseteq C$ or $\emptyset \not\models C' \sqsubseteq C_*$ and $\emptyset \models C' \sqsubseteq C$.

Cohen and Hirsh (1994a,b) and Frazier and Pitt (1996) consider concept expressions in (variations of) the now largely historic description logic Classic (Borgida et al., 1989; Patel-Schneider et al., 1991; Borgida and Patel-Schneider, 1994). The expressive power of Classic and its variants is incomparable to the expressive power of modern lightweight description logics. Classic only shares conjunction and unqualified existential restrictions of the form $\exists r.\top$ with the DLs considered in this paper. It additionally admits *value restrictions* $\forall r.C$ whose interpretation is given as

$$(\forall r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} | d' \in C^{\mathcal{I}} \text{ for all } d' \text{ with } (d, d') \in r^{\mathcal{I}}\}$$

and *unqualified number restrictions* $(\leq n\, r)$ and $(\geq n\, r)$ interpreted as

$$\begin{aligned} (\leq nr)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid |\{d' \mid (d, d') \in r^{\mathcal{I}}\}| \leq n\} \\ (\geq nr)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid |\{d' \mid (d, d') \in r^{\mathcal{I}}\}| \geq n\} \end{aligned}$$

as well as various constructors using individual names. For example, if $a_1, \ldots, a_n$ are names for individual objects, then ONE-OF$(a_1, \ldots, a_n)$ is a Classic concept denoting the set $\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$, where $a_i^{\mathcal{I}}$ denotes the individual with name a_i in interpretation $\mathcal{I}$. It is proved by Cohen and Hirsh (1994a,b) and Frazier and Pitt (1996) that in many fragments of Classic concept expressions cannot be learned polynomially using only membership or equivalence queries but that they can be learned in polynomial time using both. Exact learning of concept expressions in modern lightweight description logics has not yet been investigated.

Exact Learning of TBoxes using Concept Inclusions as Queries. First results on exact learning of description logic TBoxes using concept inclusions as queries were presented by Konev et al. (2013, 2014). This paper is an extension. In contrast to the work of Konev et al. (2013, 2014), we make the distinction between polynomial time and polynomial query learnability which enables us to formulate and prove results on a more fine grained level. TBoxes in DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$, for which we prove polynomial query learnability, were not considered by Konev et al. (2013, 2014). The current paper is also closely related to the PhD thesis of the third author (Ozaki, 2016). In addition to the results presented here, it is shown there that even in the extension of $\mathcal{EL}_{\mathsf{lhs}}$ with role inclusions, TBoxes can be learned in polynomial time. The learning algorithm is a non-trivial extension of the algorithm presented here for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes.

Exact Learning of TBoxes using Certain Answers. In recent years, data access mediated by ontologies has become one of the most important applications of DLs (Poggi et al., 2008; Bienvenu et al., 2014; Kontchakov and Zakharyaschev, 2014; Bienvenu and Ortiz, 2015) and references therein. The idea is to use a TBox to specify semantics and background knowledge for the data and use it for deriving more complete answers to queries over the data. In this context, the data is stored in an *ABox* consisting of a finite set of assertions of the form $A(a)$ or $r(a, b)$, where A is a concept names, r a role name, and a, b are individual names. Given a query $q(\vec{x})$ (typically a conjunctive query), a TBox $\mathcal{T}$, and an ABox $\mathcal{A}$, a tuple of individual names $\vec{a}$ from $\mathcal{A}$ and of the same length as $\vec{x}$ is called a certain answer to $q(\vec{x})$ over $\mathcal{A}$ w.r.t. $\mathcal{T}$, in symbols $\mathcal{T}, \mathcal{A} \models q(\vec{a})$, if every model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ satisfies $q(\vec{a})$. Motivated by this setup, Konev et al. (2016) and Ozaki (2016) study polynomial learnability of TBoxes using membership queries that ask whether a tuple of individuals names is a certain answer to a query over an ABox w.r.t. the target TBox. This is a natural alternative to learning using concept inclusions since domain experts are often more familiar with querying data in a particular domain than with the logical notion of subsumption between concept expressions. In detail, the learning protocol is as follows:

- a membership query takes the form $(\mathcal{A}, q(\vec{a}))$ and asks whether the tuple $\vec{a}$ of individual names is a certain answer to the query $q(\vec{x})$ over the ABox $\mathcal{A}$ w.r.t. the target TBox $\mathcal{T}$;

- an equivalence query asks whether a TBox $\mathcal{H}$ is equivalent to the target TBox $\mathcal{T}$. If $\mathcal{T}$ and $\mathcal{H}$ are not equivalent then a counterexample of the form $(\mathcal{A}, q(\vec{a}))$ is given such that $\mathcal{T}, \mathcal{A} \models q(\vec{a})$ and $\mathcal{H}, \mathcal{A} \not\models q(\vec{a})$ (a positive counterexample) or $\mathcal{T}, \mathcal{A} \not\models q(\vec{a})$ and $\mathcal{H}, \mathcal{A} \models q(\vec{a})$ (a negative counterexample).

In the learning protocol above we have not yet specified the class of queries from which the $q(\vec{x})$ are drawn and which strongly influences the classes of TBoxes that can be learned. In the context of data access using TBoxes the two most popular classes of queries are:

- conjunctive queries (CQs), that is, existentially quantified conjunctions of atoms; and
- instance queries (IQs), which take the form $C(x)$ or $r(x, y)$ with C a concept expression from the DL under consideration and r a role name.

Konev et al. (2016) and Ozaki (2016) study exact learning of TBoxes in the languages $\mathcal{EL}$, $\mathcal{EL}_{\mathsf{lhs}}$ and DL-Lite$_{\mathcal{R}}^{\exists}$ for both IQs and CQs in queries. The positive learnability results are proved by polynomial reductions to the learnability results presented in this paper and also by Ozaki (2016). The basic link between learning using concept inclusions as queries and learning by certain answers is as follows: if $\mathcal{T}$ is a TBox and C, D are concept expressions in any of the DLs discussed above then one can regard the labelled tree T_C corresponding to C as an ABox $\mathcal{A}_C$ with root ρ_C and it holds that $\mathcal{T} \models C \sqsubseteq D$ if, and only if, $\mathcal{T}, \mathcal{A}_C \models D(\rho_C)$. The converse direction (obtaining a concept expression from an ABox) is more involved since ABoxes are not tree-shaped and an additional unfolding step is needed to compute a corresponding concept expression. Using this link it is proved by Konev et al. (2016) and Ozaki (2016) that DL-Lite$_{\mathcal{R}}^{\exists}$ and $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes with role inclusions can be learned with polynomially many queries using certain answers to IQs. It is also proved that $\mathcal{EL}$ is still not learnable with polynomially many queries using certain answers with neither IQs nor CQs as the query language and that DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes cannot be learned with polynomially many queries using certain answers with CQs as the query language.

Exact Learning in (other) Fragments of FO Horn. We discuss results on exact learning of finite sets of FO Horn clauses or fragments of this logic, where a *FO Horn clause* is a universally quantified clause with at most one positive literal (Page Jr, 1993; Arimura, 1997; Reddy and Tadepalli, 1998; Arias and Khardon, 2002; Arias et al., 2007; Selman and Fern, 2011). Depending on what is used as membership queries and as counterexamples to equivalence queries, one can distinguish between exact learning FO Horn clauses using interpretations and using entailments. As learning using entailments is closer to our approach we focus on that setting. The exact learning protocol is then as follows:

- a membership query asks whether an FO Horn clause is entailed by the target set T of FO Horn clauses;
- an equivalence query asks whether a set H of FO Horn clauses is equivalent to the target set T. If H and T are not equivalent then a counterexample is given, that is, an FO Horn clause entailed by T but not by H (a positive counterexample) or vice versa.

Considering how terms (with function symbols allowed) can appear in an FO Horn clause, two main restrictions have been studied in the literature:

1. *Range restricted clauses*: when the set of terms in the positive literal (if existent) is a subset of the terms in the negative literals and their subterms; and
2. *Constrained clauses*: when the set of terms and subterms in the positive literal (if existent) is a superset of the terms in the negative literals.

For example, the FO Horn clause $\forall x(\neg P(f(x)) \vee P(x))$ is range restricted but not constrained and the FO Horn clause $\forall x(\neg P(x) \vee P(f(x)))$ is constrained but not range restricted, where P is a predicate symbol and f a function symbol. Reddy and Tadepalli (1998) and Arimura (1997), it is shown that under certain acyclicity conditions FO Horn with range restricted clauses and, respectively, constrained clauses are polynomial time learnable from entailments if the arity of predicates is bounded by a constant. A learning algorithm for a fragment of FO Horn (called closed FO Horn) that subsumes the two languages defined above is presented by Arias and Khardon (2002). The algorithm is polynomial in the number of clauses, terms and predicates and the size of the counterexamples, but exponential not only in the arity of predicates but also in the number of variables per clause. In fact, it is an open question whether there exists a learning algorithm for closed FO Horn that is polynomial in the number of variables per clause.

We relate the learnability results for FO Horn to the learnability results for lightweight description logics presented in this paper. Observe that most DLs (and in particular all DLs investigated in this paper) can be translated into FO (Baader et al., 2003). For example, a translation of the $\mathcal{EL}_{\mathsf{lhs}}$ CI $\exists r.A \sqsubseteq B$ is $\forall x \forall y(\neg r(x,y) \vee \neg A(y) \vee B(x))$ and a translation of the DL-Lite$^{\exists}_{\mathcal{R}}$ CI $A \sqsubseteq \exists r.A$ is $\forall x(A(x) \rightarrow \exists y.(r(x,y) \wedge A(y)))$. Under this translation, every $\mathcal{EL}_{\mathsf{lhs}}$ TBox can be regarded as a set of range restricted FO Horn clauses, where the arity of predicates is bounded by 2. In contrast, since in DL-Lite$^{\exists}_{\mathcal{R}}$ existential quantifiers can be nested in the right side of CIs, DL-Lite$^{\exists}_{\mathcal{R}}$ CIs cannot be translated into FO Horn clauses. We can now summarise the relationship between our learnability results for $\mathcal{EL}_{\mathsf{lhs}}$, DL-Lite$^{\exists}_{\mathcal{R}}$ and DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$ and the results on exact learnability of FO Horn from entailments as follows: since the arity of DL predicates is at most 2 and since no function symbols are admitted in DLs, none of the DLs considered in this paper can express the fragments of FO Horn discussed above. On the other hand, we do not impose an acyclicity condition on the TBoxes (in contrast to the work by Reddy and Tadepalli (1998); Arimura (1997)) and our algorithms are polynomial in the number of variables permitted in any clause (in contrast to the work by Arias and Khardon (2002)). Thus, the results discussed above for FO Horn do not translate into polynomial learning algorithms for $\mathcal{EL}_{\mathsf{lhs}}$ and are not applicable to DL-Lite$^{\exists}_{\mathcal{R}}$ nor DL-Lite$^{\exists}_{\mathcal{R},\mathsf{horn}}$. Our results thus cover new fragments of FO that have not yet been considered for exact learning. This is not surprising, given the fact that the fragments of FO considered previously were not motivated by applications in ontology learning.

Also related to exact learning of Horn FO is recent work on exact learning of schema mappings in data exchange (ten Cate et al., 2012). Schema mappings are tuples (S, T, M) where S is a source schema (a finite set of predicates), T is a target schema (a finite set of predicates), and M is a finite set of sentences of the form $\forall \vec{x}(\varphi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y}))$ where $\varphi(\vec{x})$ and $\psi(\vec{x}, \vec{y})$ are conjunctions of atoms over S and T, respectively (Fagin et al., 2005). (S, T, M) is a GAV schema mapping if $\vec{y}$ is empty and $\psi(\vec{x}, \vec{y})$ is an atom. The authors study exact learnability of GAV schema mappings from data examples (I, J) consisting of a database I over the source schema S and a database J over the target schema T. Such a data example satisfies M if $I \cup J \models M$. The authors present both polynomial query learnability results for protocols using membership and equivalence queries and non-polynomial query learnability results if either only membership or only equivalence queries are allowed. These results are not applicable to the setting considered in this paper since the learning protocol uses data examples instead of entailments.

8. Conclusion

We have presented the first study of learnability of DL ontologies in Angluin et al's framework of exact learning, obtaining both positive and negative results. Several research questions remain to be explored. One immediate question is whether acyclic $\mathcal{EL}$ TBoxes can be learned in polynomial time using queries and counterexamples of the form $A \equiv C$ and $A \sqsubseteq C$ only. Note that our non-polynomial query learnability result for acyclic $\mathcal{EL}$ TBoxes relies heavily on counterexamples that are not of this form. Another immediate question is whether the extension of $\mathcal{EL}_{\mathsf{lhs}}$ with inverse roles (which is a better approximation of OWL2 RL than $\mathcal{EL}_{\mathsf{lhs}}$ itself) can still be learned in polynomial time, or at least with polynomially many queries of polynomial size. Other interesting research directions are non-polynomial time learning algorithms for $\mathcal{EL}$ TBoxes and the admission of different types of membership queries and counterexamples in the learning protocol. For example, one could replace CIs as counterexamples with interpretations.

Acknowledgments

We would like to thank the reviewers for their helpful comments. Lutz was supported by the DFG project Prob-DL (LU1417/1-1). Konev and Wolter were supported by the EPSRC project EP/H043594/1. Ozaki was supported by the Science without Borders scholarship programme (245288/2012-0) and cfaed.

Appendix A. Proofs for Section 6

We supply proofs for Lemma 62 and Lemma 64. In addition, we prove a claim used in the proof of Lemma 67. We start by giving the proof of Lemma 62.

Lemma 62 *For every DL-Lite$^{\exists}_{\mathcal{R}}$ CI $B \sqsubseteq D$ over the signature of $\mathcal{T}_{\boldsymbol{\sigma}}$,*

- *either $\mathcal{T}_{\boldsymbol{\sigma}} \models B \sqsubseteq D$ for every $\mathcal{T}_{\boldsymbol{\sigma}} \in S_N$*
- *or there is at most one $\mathcal{T}_{\boldsymbol{\sigma}} \in S_N$ such that $\mathcal{T}_{\boldsymbol{\sigma}} \models B \sqsubseteq D$.*

Proof Assume the CI $B \sqsubseteq D$ is given. If $B \neq A$ or M does not occur in D, then the claim can be readily checked. Thus, we assume that $B = A$ and M occurs in D. Assume there exists $\boldsymbol{\sigma}_0$ such that $\mathcal{T}_{\boldsymbol{\sigma}_0} \models A \sqsubseteq D$ (if no such $\boldsymbol{\sigma}_0$ exists, we are done). For any $\boldsymbol{\sigma}$, let $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}}}$ be the canonical model of A and $\mathcal{T}_{\boldsymbol{\sigma}}$ (Lemma 12). Apply the following restricted form of parent/child merging exhaustively to the concept expression D:

- if there are nodes $d, d_1, d_2 \in T_D$ with $l(d_1, d) = \sigma$ and $l(d, d_2) = \sigma^-$ for some $\sigma \in \{r, s\}$, then replace D by the resulting concept expression after d_1 and d_2 are merged in D.

Let D' be the resulting concept expression. Recall from Lemma 12 that $\mathcal{T}_{\boldsymbol{\sigma}} \models A \sqsubseteq D$ iff there is a homomorphism from T_D to $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}}}$ mapping ρ_D to ρ_A. Using the fact that $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}}}$ is a ditree interpretation, one can readily check that any homomorphism h from T_D to $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}}}$ mapping ρ_D to ρ_A factors through $T_{D'}$ and that D' is an $\mathcal{EL}$ concept expression. Thus, if there is an additional $\boldsymbol{\sigma}' \neq \boldsymbol{\sigma}_0$ such that $\mathcal{T}_{\boldsymbol{\sigma}'} \models A \sqsubseteq D$, then there are two homomorphisms

$h_{\boldsymbol{\sigma}_0}$ and $h_{\boldsymbol{\sigma}'}$ with the same domain $T_{D'}$ into $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}_0}}$ and $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}'}}$ and mapping the root of $T_{D'}$ to the roots of $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}_0}}$ and $\mathcal{I}_{A,\mathcal{T}_{\boldsymbol{\sigma}'}}$, respectively. Since M occurs in D' and D' is an $\mathcal{EL}$ concept expression we find a sequence $D_0, \ldots, D_m$ with $D_0 = D'$ and $D_m = M$ such that $\exists s_{i+1}.D_{i+1}$ is a top-level conjunct of D_i for $s_i \in \{r, s\}$ and all $i < m$. But then $s_1 \cdots s_m = \boldsymbol{\sigma}_0$ and $s_1 \cdots s_m = \boldsymbol{\sigma}'$ and we have derived a contradiction to the assumption that $\boldsymbol{\sigma}_0$ and $\boldsymbol{\sigma}'$ are distinct. ❑

To prove Lemma 64 we require the following observation.

Lemma 72 *For any acyclic $\mathcal{EL}$ TBox $\mathcal{T}$, any CI $A \sqsubseteq C \in \mathcal{T}$ and any concept expression of the form $\exists t.D$ we have $\mathcal{T} \models A \sqsubseteq \exists t.D$ if, and only if, $\mathcal{T} \models C \sqsubseteq \exists t.D$.*

We are now ready to prove Lemma 64.

Lemma 64 *For every $\mathcal{EL}$ CI $C \sqsubseteq D$ over Σ_n:*

- *either $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$*
- *or the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq D$ does not exceed $|C|$.*

Proof We prove the lemma by induction on the structure of D. We assume throughout the proof that there exists some $L_0 \in \mathfrak{L}_n$ such that $\mathcal{T}_{L_0} \models C \sqsubseteq D$.

Base case: D is a concept name. We make the following case distinction.

- $D \in \{X_i, A_i, B_i \mid 1 \leq i \leq n\}$ or $D = M$. By Lemma 65, C is of the form $Z \sqcap C'$, for some concept name Z, and $\mathcal{T}_{L_0} \models Z \sqsubseteq D$. But then $Z = D$ and it follows that $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$.
- $D = X_0$. By Lemma 65, C is of the form $Z \sqcap C'$, for some concept name Z, and $\mathcal{T}_{L_0} \models Z \sqsubseteq X_0$. This is the case if either $Z = X_0$, or $Z \in \{A, A_1, B_1, \ldots, A_n, B_n\}$. In either case, $\mathcal{T}_L \models C \sqsubseteq X_0$ for every $L \in \mathfrak{L}_n$.
- $D = A$. If C is of the form $A \sqcap C'$ or for all i such that $1 \leq i \leq n$, A_i or B_i is a conjunct of C, then $\mathcal{T}_L \models C \sqsubseteq A$ for every $L \in \mathfrak{L}_n$. Assume now that C is not of this form. Then for some j such that $1 \leq j \leq n$, C is neither of the form $A \sqcap C'$ nor of the form $A_j \sqcap C'$ nor of the form $B_j \sqcap C'$. Let $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n) \in \mathfrak{L}_n$ be such that $\mathcal{T}_L \models C \sqsubseteq A$. Notice that $\mathcal{T}_L \models C \sqsubseteq A$, for $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n) \in \mathfrak{L}_n$, if, and only if, $\mathcal{T}_L \models C \sqsubseteq X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \cdots \sqcap \exists \boldsymbol{\sigma}_n.M$. By the claim in the proof of Lemma 66, for such a $\mathcal{T}_L$ we must have $\emptyset \models C \sqsubseteq \exists \boldsymbol{\sigma}_j.M$. Clearly, the number of $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n) \in \mathfrak{L}_n$ with $\emptyset \models C \sqsubseteq \exists \boldsymbol{\sigma}_j.M$ does not exceed $|C|$.

  Thus, either $\mathcal{T}_L \models C \sqsubseteq A$ for every $L \in \mathfrak{L}_n$ or the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq A$ does not exceed $|C|$.

Induction step. If $D = D_1 \sqcap D_2$, then $\mathcal{T}_L \models C \sqsubseteq D$ if, and only if, $\mathcal{T} \models C \sqsubseteq D_i$, $i = 1, 2$. By induction hypothesis, for $i = 1, 2$ either $\mathcal{T}_L \models C \sqsubseteq D_i$ for every $L \in \mathfrak{L}_n$ or there exist at most $|C|$ different $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq D_i$. Thus either $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$ or the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq D$ also does not exceed $|C|$.

Now assume that $D = \exists t.D'$. Suppose that $\mathcal{T}_L \models C \sqsubseteq D$ for some $L \in \mathfrak{L}_n$. Then, by Lemma 65, either there exists a conjunct Z of C, Z a concept name, such that $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$

or there exists a conjunct $\exists t.C'$ of C with $\mathcal{T}_L \models C' \sqsubseteq D'$. We analyse for every conjunct of C of the form Z or $\exists t.C'$ the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$ or $\mathcal{T}_L \models \exists t.C' \sqsubseteq \exists t.D'$, respectively.

(i) Let Z be a conjunct of C such that Z is a concept name and $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$. Notice that $Z \neq M$ as there is no $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models M \sqsubseteq \exists t.D'$. We consider the remaining cases.

- $Z = X_i$, for some $i \geq 0$. It is easy to see that for $L, L' \in \mathfrak{L}_n$ we have $\mathcal{T}_L \models X_i \sqsubseteq \exists t.D'$ if, and only if $\mathcal{T}_{L'} \models X_i \sqsubseteq \exists t.D'$. Thus, $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$ for every $L \in \mathfrak{L}_n$.
- $Z \in \{A_i, B_i \mid 1 \leq i \leq n\}$. By Lemma 72, $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$ if, and only if, $\mathcal{T}_L \models X_0 \sqcap \exists \boldsymbol{\sigma}_i.M \sqsubseteq \exists t.D'$. By Lemma 65, either $\mathcal{T}_L \models X_0 \sqsubseteq \exists t.D'$ or $\mathcal{T}_L \models \exists \boldsymbol{\sigma}_i.M \sqsubseteq \exists t.D'$. If $\mathcal{T}_L \models X_0 \sqsubseteq \exists t.D'$ then, as above, for $\mathcal{T}_L \models C \sqsubseteq \exists t.D'$ every $L \in \mathfrak{L}_n$. Suppose that $\exists t.D'$ is such that $\mathcal{T}_L \not\models X_0 \sqsubseteq \exists t.D'$ and $\mathcal{T}_L \models \exists \boldsymbol{\sigma}_i.M \sqsubseteq \exists t.D'$. By inductive applications of Lemma 65, this is only possible if $\exists t.D' = \exists \boldsymbol{\sigma}_i.M$. Thus, there is exactly one $L \in \mathfrak{L}_n$ (namely, $L = L_0$) such that $\mathcal{T}_L \models Z \sqsubseteq \exists \boldsymbol{\sigma}_i.M$.
- $Z = A$. Suppose that for some $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n) \in \mathfrak{L}_n$ we have $\mathcal{T}_L \models A \sqsubseteq \exists t.D'$. Equivalently, $\mathcal{T}_L \models X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \ldots \boldsymbol{\sigma}_n.M \sqsubseteq \exists t.D'$. By Lemma 65, either $\mathcal{T}_L \models X_0 \sqsubseteq \exists t.D'$ or $\mathcal{T}_L \models \exists \boldsymbol{\sigma}_i.M \sqsubseteq \exists t.D'$ for some i with $1 \leq i \leq n$. Thus, as above, unless $\mathcal{T}_L \models X_0 \sqsubseteq \exists t.D'$ we have $\exists t.D'$ is $\exists \boldsymbol{\sigma}_i.M$. But then $L = L_0$.

(ii) Let $\exists t.C'$ be a conjunct of C with $\mathcal{T}_L \models C' \sqsubseteq D'$. The induction hypothesis implies that the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C' \sqsubseteq D'$ does not exceed $|C'|$.

To summarise, either $\mathcal{T}_L \models C \sqsubseteq \exists t.D'$ for every $L \in \mathfrak{L}_n$ or for every conjunct C_0 of C of the form Z or $\exists t.C'$, the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C_0 \sqsubseteq \exists t.D'$ does not exceed $|C_0|$. Hence the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq \exists t.D'$ does not exceed $|C|$. ❏

The next result is used in the proof of Lemma 67.

Lemma 73 *For any $0 \leq i \leq n$ and concept expression D over Σ_n, if $\mathcal{T}_0 \not\models X_i \sqsubseteq D$ then there exists a sequence of role names $t_1, \ldots t_l \in \{r, s\}^l$ such that $\emptyset \models D \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ and $\mathcal{T}_0 \not\models X_i \sqsubseteq \exists t_1. \cdots \exists t_l.Y$, where Y is either $\top$ or a concept name and $0 \leq l \leq n - i + 1$.*

Proof We prove the lemma by induction on i from $i = n$ to $i = 0$. If $i = n$, then $\mathcal{T}_0 \not\models X_i \sqsubseteq D$ if either $\emptyset \models D \sqsubseteq \exists t.\top$, for some role name t, or $\emptyset \models D \sqsubseteq Y$, for some concept name $Y \neq X_i$.

Suppose that the lemma is proved for $0 < j \leq n$ and let $i = j-1$. We proceed by induction on the structure of D. If D is a concept name, we are done as $\mathcal{T}_0 \models X_i \sqsubseteq Z$ does not hold for any concept name $Z \neq X_i$. If D is of the form $\exists t.D'$, where $t \in \{r, s\}$, then $\mathcal{T}_0 \not\models X_{i+1} \sqsubseteq D'$, and so, by induction hypothesis, there exists a sequence of role names $t_1, \ldots, t_l$, with $l \leq n-i$, such that $\mathcal{T}_0 \not\models X_{i+1} \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ and $\emptyset \models D' \sqsubseteq \exists t_1. \cdots \exists t_l.Y$. But then, by Lemma 72 and Lemma 65, $\mathcal{T}_0 \not\models X_i \sqsubseteq \exists t.\exists t_1. \cdots \exists t_l.Y$ and $\emptyset \models \exists t.D' \sqsubseteq \exists t.\exists t_1. \cdots \exists t_l.Y$. If D is of the form $D = D_1 \sqcap D_2$, there there exists D_i, $i = 1, 2$, such that $\mathcal{T}_0 \not\models X_i \sqsubseteq D_i$ and the lemma holds by induction hypothesis. ❏

References

Dana Angluin. Learning propositional Horn sentences with hints. Technical report, Yale University, 1987a.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987b.

Dana Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

Dana Angluin, Michael Frazier, and Leonard Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.

Marta Arias. *Exact learning of first-order expressions from queries*. PhD thesis, Tufts University, 2004.

Marta Arias and José L. Balcázar. Construction and learnability of canonical Horn formulas. *Machine Learning*, 85(3):273–297, 2011.

Marta Arias and Roni Khardon. Learning closed Horn expressions. *Information and Computation*, 178(1):214–240, 2002.

Marta Arias, Roni Khardon, and Jérôme Maloberti. Learning Horn expressions with LOGAN-H. *Journal of Machine Learning Research*, 8:549–587, 2007.

Hiroki Arimura. Learning acyclic first-order Horn sentences from entailment. In *International Workshop on Algorithmic Learning Theory*, pages 432–445, 1997.

Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)*, 36:1–69, 2009.

Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 96–103, 1999.

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003. ISBN 0-521-78176-0.

Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 364–369, 2005.

Franz Baader, Bernhard Ganter, Baris Sertkaya, and Ulrike Sattler. Completing description logic knowledge bases using formal concept analysis. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 230–235, 2007.

Franz Baader, Carsten Lutz, and Sebastian Brandt. Pushing the EL envelope further. In *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*, 2008.

Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web. Semantic Technologies for Advanced Query Answering - 11th International Summer School*, pages 218–307, 2015.

Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

Daniel Borchmann and Felix Distel. Mining of $\mathcal{EL}$-GCIs. In *The 11th IEEE International Conference on Data Mining Workshops*, Vancouver, Canada, 2011.

Daniel Borchmann. *Learning terminological knowledge with high confidence from erroneous data*. PhD thesis, Higher School of Economics, 2014.

Alexander Borgida and Peter F Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research*, 1: 277–308, 1994.

Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 58–67, 1989.

Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

Lorenz Bühmann, Daniel Fleischhacker, Jens Lehmann, André Melo, and Johanna Völker. Inductive lexical learning of class expressions. In *Knowledge Engineering and Knowledge Management - 19th International Conference (EKAW)*, pages 42–53, 2014.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated reasoning*, 39(3):385–429, 2007.

Philipp Cimiano, Johanna Völker, and Paul Buitelaar. Ontology construction. In *Handbook of Natural Language Processing, Second Edition.*, pages 577–604. Chapman and Hall/CRC, 2010.

William W Cohen and Haym Hirsh. The learnability of description logics with equality constraints. *Machine Learning*, 17(2-3):169–199, 1994a.

William W. Cohen and Haym Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In *Principles of Knowledge Representation and Reasoning (KR)*, pages 121–133, 1994b.

Felix Distel. *Learning description logic knowledge bases from data using methods from formal concept analysis*. PhD thesis, Dresden University of Technology, 2011.

Ronald Fagin, Phokion G Kolaitis, Renée J Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

Daniel Fleischhacker, Johanna Völker, and Heiner Stuckenschmidt. Mining RDF data for property axioms. In *On the Move to Meaningful Internet Systems: OTM 2012*, pages 718–735. Springer, 2012.

Michael Frazier and Leonard Pitt. Learning from entailment: An application to propositional Horn sentences. In *International Conference on Machine Learning (ICML)*, pages 120–127, 1993.

Michael Frazier and Leonard Pitt. Classic learning. *Machine Learning*, 25(2-3):151–193, 1996.

Bernhard Ganter, Sergei A Obiedkov, Sebastian Rudolph, and Gerd Stumme. *Conceptual exploration*. Springer, 2016.

Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, Ian Horrocks, Christoph Pinkel, Martin G Skjæveland, Evgenij Thorstensen, and Jose Mora. BootOX: practical mapping of RDBs to OWL 2. In *International Semantic Web Conference, (ISWC)*, pages 113–132. 2015.

Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev. On (in)tractability of OBDA with OWL 2 QL. In *Proceedings of the 24th International Workshop on Description Logics (DL 2011)*, 2011.

Szymon Klarman and Katarina Britz. Ontology learning from interpretations in lightweight description logics. In *Inductive Logic Programming*, 2015.

Boris Konev, Michel Ludwig, Dirk Walther, and Frank Wolter. The logical difference for the lightweight description logic EL. *Journal of Artificial Intelligence Research (JAIR)*, 44: 633–708, 2012.

Boris Konev, Carsten Lutz, and Frank Wolter. Exact learning of Tboxes in EL and DL-Lite. In *Informal Proceedings of the 26th International Workshop on Description Logics*, pages 341–352, 2013.

Boris Konev, Carsten Lutz, Ana Ozaki, and Frank Wolter. Exact learning of lightweight description logic ontologies. In *Principles of Knowledge Representation and Reasoning (KR)*, 2014.

Boris Konev, Ana Ozaki, and Frank Wolter. A model for learning description logic ontologies based on exact learning. In *Conference on Artificial Intelligence (AAAI)*, pages 1008–1015, 2016.

Roman Kontchakov and Michael Zakharyaschev. An introduction to description logics and query rewriting. In *Reasoning Web. Semantic Technologies for Advanced Query Answering - 10th International Summer School*, pages 195–244, 2014.

Markus Krötzsch. OWL 2 profiles: An introduction to lightweight ontology languages. In *Reasoning Web. Semantic Technologies for Advanced Query Answering - 8th International Summer School*, pages 112–183, 2012.

Jens Lehmann and Christoph Haase. Ideal downward refinement in the EL description logic. In *International Conference on Inductive Logic Programming (ILP)*, pages 73–87, 2009.

Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203–250, 2010.

Jens Lehmann and Johanna Völker. *Perspectives on Ontology Learning*, volume 18. IOS Press, 2014.

Francesca A. Lisi. Al-quin: An onto-relational learning system for semantic web mining. *International Journal on Semantic Web and Information Systems*, 7(3):1–22, 2011.

Francesca A. Lisi and Umberto Straccia. Learning in description logics with fuzzy concrete domains. *Fundamenta Informaticae*, 140(3-4):373–391, 2015.

Carsten Lutz, Robert Piro, and Frank Wolter. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 983–988, 2011.

Yue Ma and Felix Distel. Learning formal definitions for snomed ct from text. In *Artificial Intelligence in Medicine Conference (AIME)*, pages 73–77, 2013.

Ana Ozaki. *Exact Learning of Description Logic Ontologies*. PhD thesis, University of Liverpool, 2016.

Charles David Page Jr. *Anti-unification in constraint logics: foundations and applications to learnability in first-order logic, to speed-up learning, and to deduction*. PhD thesis, University of Illinois at Urbana-Champaign, 1993.

Peter F. Patel-Schneider, Deborah L. McGuinness, and Alexander Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rationale. *SIGART Bulletin*, 2(3):108–113, 1991.

Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10: 133–173, 2008.

Chandra Reddy and Prasad Tadepalli. Learning first-order acyclic Horn programs from entailment. *Inductive Logic Programming*, pages 23–37, 1998.

Sebastian Rudolph. Exploring relational structures via FLE. In *International Conference on Conceptual Structures, ICCS*, pages 196–212, 2004.

Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank Van Harmelen. Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39(3):317–349, 2007.

Joseph Selman and Alan Fern. Learning first-order definite theories via object-based queries. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 159–174, 2011.

Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer, 2009.

Balder ten Cate, Víctor Dalmau, and Phokion G. Kolaitis. Learning schema mappings. In *International Conference on Database Theory (ICDT)*, pages 182–195, 2012.

Johanna Völker and Mathias Niepert. Statistical schema induction. In *The Semantic Web: Research and Applications*, pages 124–138. Springer, 2011.

Johanna Völker, Daniel Fleischhacker, and Heiner Stuckenschmidt. Automatic acquisition of class disjointness. *Journal of Web Semantics*, 35:124–139, 2015.

Hai Wang, Matthew Horridge, Alan L. Rector, Nick Drummond, and Julian Seidenberg. Debugging OWL-DL ontologies: A heuristic approach. In *4th International Semantic Web Conference ISWC*, pages 745–757, 2005.

Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology learning from text: A look back and into the future. *ACM Computing Surveys*, 44(4):20:1–20:36, 2012.