A Model for Learning Description Logic Ontologies Based on Exact Learning

Boris Konev
University of Liverpool
United Kingdom

Ana Ozaki
University of Liverpool
United Kingdom

Frank Wolter
University of Liverpool
United Kingdom

Abstract

We investigate the problem of learning description logic (DL) ontologies in Angluin et al.'s framework of exact learning via queries posed to an oracle. We consider membership queries of the form "is a tuple $\vec{a}$ of individuals a certain answer to a data retrieval query q in a given ABox and the unknown target ontology?" and completeness queries of the form "does a hypothesis ontology entail the unknown target ontology?". Given a DL L and a data retrieval query language Q, we study polynomial learnability of ontologies in L using data retrieval queries in Q and provide an almost complete classification for DLs that are fragments of $\mathcal{EL}$ with role inclusions and of DL-Lite and for data retrieval queries that range from atomic queries and $\mathcal{EL}/\mathcal{ELI}$-instance queries to conjunctive queries. Some results are proved by non-trivial reductions to learning from subsumption examples.

Introduction

Building an ontology is prone to errors, time consuming, and costly. The research community has addressed this problem in many different ways, for example, by supplying tool support for editing ontologies (Musen 2013; Bechhofer et al. 2001; Day-Richter et al. 2007), developing reasoning support for debugging ontologies (Wang et al. 2005; Schlobach et al. 2007), supporting modular ontology design (Stuckenschmidt, Parent, and Spaccapietra 2009), and by investigating automated ontology generation from data or text (Cimiano, Hotho, and Staab 2005; Buitelaar, Cimiano, and Magnini 2005; Lehmann and Völker 2014; Borchmann and Distel 2011; Ma and Distel 2013). One major problem when building an ontology is the fact that domain experts are rarely ontology engineering experts and that, conversely, ontology engineers are typically not familiar with the domain of the ontology. An ontology building project therefore often relies on the successful communication between an ontology engineer (familiar with the semantics of ontology languages) and a domain expert (familiar with the domain of interest). In this paper, we consider a simple model of this communication process and analyse, within this model, the computational complexity of reaching a correct and complete domain ontology. We assume that

- the domain expert knows the domain ontology and its vocabulary without being able to formalize or communicate this ontology;
- the domain expert is able to communicate the vocabulary of the ontology and shares it with the ontology engineer. Thus, the domain expert and ontology engineer have a common understanding of the vocabulary of the ontology. The ontology engineer knows nothing else about the domain.
- the ontology engineer can pose queries to the domain expert which the domain expert answers truthfully. Assuming that the domain expert can interpret data in her area of expertise, the main queries posed by the ontology engineer are based on data retrieval examples:
  - assume a data instance $\mathcal{A}$ and a data retrieval query $q(\vec{x})$ are given. Is the tuple $\vec{a}$ of individuals a certain answer to query $q(\vec{x})$ in $\mathcal{A}$ and the ontology $\mathcal{O}$?

  In addition, we require a way for the ontology engineer to find out whether she has reconstructed the target ontology already and, if this is not the case, to request an example illustrating the incompleteness of the reconstruction. We abstract from defining a communication protocol for this, but assume for simplicity that the following query can be posed by the ontology engineer:

  - Is this ontology $\mathcal{H}$ complete? If not, return a data instance $\mathcal{A}$, a query $q(\vec{x})$, and a tuple $\vec{a}$ such that $\vec{a}$ is a certain answer to $q(\vec{x})$ in $\mathcal{A}$ and the ontology $\mathcal{O}$ and is not a certain answer to $q(\vec{x})$ in $\mathcal{A}$ and the ontology $\mathcal{H}$.

Given this model, our question is whether the ontology engineer can learn the target ontology $\mathcal{O}$ and which computational resources are required for this depending on the ontology language in which the ontology $\mathcal{O}$ and the hypothesis ontology $\mathcal{H}$ are formulated. Our model obviously abstracts from a number of fundamental problems in building ontologies and communicating about them. In particular, it makes the assumption that the domain expert knows the domain ontology and its vocabulary (without being able to formalize it) despite the fact that finding an appropriate vocabulary for a domain of interest is a major problem in ontology design (Lehmann and Völker 2014). We make this assumption here in order to isolate the problem of communication about the logical relationships between known vocabulary items and its dependence on the ontology language within which the relationships can be formulated.

The model described above is an instance of Angluin et al.'s framework of exact learning via queries to an oracle (Angluin 1987). The queries using data retrieval examples can be regarded as membership queries posed by a learner to an oracle and the completeness query based on a hypothesis $\mathcal{H}$ can be regarded as an equivalence query by the learner to the oracle. Formulated in Angluin's terms we are thus interested in whether there exists a deterministic learning algorithm that poses membership and equivalence queries of the above form to an oracle and that polynomially learns an arbitrary ontology over a given ontology language.

As usual in the exact learning literature, we consider two distinct notions of polynomial learnability: polynomial *time* learnability and polynomial *query* learnability. If one can learn TBoxes[1] in a given DL L with a deterministic algorithm using polynomially many polynomial size queries, then we say that TBoxes in L are polynomial query learnable. If one can learn TBoxes in L with a deterministic algorithm in polynomial time, then we say that TBoxes in L are polynomial time learnable. Precise definitions are given below. Clearly, polynomial time learnability implies polynomial query learnability. The converse does not hold for arbitrary learning problems. Intuitively, when studying polynomial time learnability one takes into account potentially costly transformations of counterexamples to equivalence queries provided by the oracle that the learning algorithm is required to do when it analyses the counterexamples. In contrast, when studying polynomial query learnability one abstracts from such intermediate computations and focuses on the learning protocol itself. It turns out that for the DLs considered in this paper in many cases there is no difference between polynomial time and polynomial query learnability; the only exception, however, is rather interesting and will be discussed in detail below.

We investigate polynomial learnability for seven DLs and four query languages: the DLs are $\mathcal{EL}$ and its fragments $\mathcal{EL}_{\mathsf{lhs}}$ and $\mathcal{EL}_{\mathsf{rhs}}$ in which complex concepts are allowed only on the left-hand and, respectively, right-hand side of concept inclusions. We also consider their extensions $\mathcal{ELH}$, $\mathcal{ELH}_{\mathsf{lhs}}$, and $\mathcal{ELH}_{\mathsf{rhs}}$ with role inclusions. In addition, we consider the DL-Lite dialect DL-Lite$^{\exists}_{\mathcal{H}}$ which is defined as the extension of $\mathcal{ELH}_{\mathsf{rhs}}$ with inverse roles. We thus consider significant fragments of the OWL2 EL and OWL2 QL profiles of the web ontology language OWL. The introduction of the fragments $\mathcal{EL}_{\mathsf{lhs}}$ and $\mathcal{EL}_{\mathsf{rhs}}$ is motivated by the fact that $\mathcal{EL}$ TBoxes typically cannot be polynomially learned (see below). In data retrieval examples we consider the following standard query languages: atomic queries (AQs), $\mathcal{EL}$-instance queries ($\mathcal{EL}$-IQs), $\mathcal{ELI}$-instance queries ($\mathcal{ELI}$-IQs), and conjunctive queries (CQs).

Our results regarding polynomial *query* learnability of TBoxes are presented in Table 1. In the table, $\mathcal{EL}(\mathcal{H})$ ranges over $\mathcal{EL}$ and $\mathcal{ELH}$ and (–) denotes that the query language is not expressive enough to determine a unique (up to logical equivalence) TBox in the corresponding DL using data retrieval examples. Thus, in those cases *no* learning algorithm exists, whereas in all other cases one can easily construct a learning algorithm that makes exponentially many queries. Note that the table shows that for the $\mathcal{EL}$-dialects polynomial query learnability does not depend on whether role inclusions are present (though some proofs are considerably harder with role inclusions). A particularly interesting result is that $\mathcal{EL}_{\mathsf{rhs}}$ TBoxes are polynomially query learnable using IQs in data retrieval examples but not using CQs. Thus, a more expressive language for communication does not always lead to more efficient communication.

The bottom row shows polynomial query learnability results for the case in which *concept subsumptions* rather than data retrieval examples are used in the communication between the learner and the oracle. Except for polynomial query learnability of $\mathcal{ELH}_{\mathsf{lhs}}$ (which we prove in this paper), the results for subsumption are from (Konev et al. 2014).[2] Our polynomial query learnability results for data retrieval examples are by reductions to learnability using concept subsumptions. Our focus on data retrieval examples rather than subsumptions is motivated by the observation that domain experts are often more familiar with querying data in their domain than with the logical notion of subsumption between complex concepts.

We now discuss our results for polynomial *time* learnability. As mentioned above, all non polynomial query learnability results transfer to non polynomial time learnability results. Moreover, in both the subsumption and the data retrieval frameworks our proofs of positive polynomial query learnability results for $\mathcal{EL}_{\mathsf{lhs}}$, $\mathcal{ELH}_{\mathsf{lhs}}$, $\mathcal{EL}_{\mathsf{rhs}}$, and $\mathcal{ELH}_{\mathsf{rhs}}$ actually prove polynomial time learnability. In fact, the only case in which we have not been able to extend a polynomial query learnability result to a polynomial time learnability result is for DL-Lite$^{\exists}_{\mathcal{H}}$ TBoxes: it remains open whether DL-Lite$^{\exists}_{\mathcal{H}}$ TBoxes can be learned in polynomial time using subsumption or $\mathcal{ELI}$-IQs in data retrieval queries. The reason is interesting: checking whether an $\mathcal{ELI}$-IQ is entailed by a DL-Lite$^{\exists}_{\mathcal{H}}$ TBox and ABox is NP-complete in combined complexity (Kikot, Kontchakov, and Zakharyaschev 2011) and such entailment checks are required in our polynomial query learning algorithm to transform counterexamples provided by the oracle. It remains open whether our learning algorithm can be modified in such a way that no such entailment checks are required. In contrast to DL-Lite$^{\exists}_{\mathcal{H}}$, in $\mathcal{ELH}_{\mathsf{rhs}}$ the corresponding entailment problem is in PTime in combined complexity (Bienvenu et al. 2013), and so a polynomial time learning algorithm can use entailment checks.

Finally, we note that the two open problems in Table 1 for

[1]In the DL context we identify *ontologies* with TBoxes.

[2]The authors of (Konev et al. 2014) consider polynomial time learnability only. As polynomial time learnability implies polynomial query learnability, the corresponding results in Table 1 follow. Note that the learning algorithm for DL-Lite$^{\exists}_{\mathcal{H}}$ TBoxes given in (Konev et al. 2014) only shows polynomial query learnability of DL-Lite$^{\exists}_{\mathcal{H}}$ TBoxes using subsumption queries but does not show polynomial time learnability of DL-Lite$^{\exists}_{\mathcal{H}}$ TBoxes using subsumption queries (it is wrongly assumed that checking $\mathcal{T} \models C \sqsubseteq D$ is in PTime for DL-Lite$^{\exists}_{\mathcal{H}}$ TBoxes $\mathcal{T}$ and concept inclusions $C \sqsubseteq D$). In fact, polynomial time learnability of DL-Lite$^{\exists}_{\mathcal{H}}$ TBoxes using subsumption queries is an open problem (see below). All other polynomial time learnability results in (Konev et al. 2014) hold.

Table 1: Positive (✓) and negative (✗) results regarding polynomial query learnability.

Framework		$\mathcal{EL}(\mathcal{H})_{\mathsf{lhs}}$	$\mathcal{EL}(\mathcal{H})_{\mathsf{rhs}}$	$\mathcal{EL}(\mathcal{H})$	DL-Lite$^{\exists}_{\mathcal{H}}$
Data	AQs	✓	–	–	–
	$\mathcal{EL}$-IQs	✓	✓	✗	–
	$\mathcal{ELI}$-IQs	✓	✓	?	✓
	CQs	✓	✗	✗	?
Subsump.		✓	✓	✗	✓

polynomial query learnability are open for polynomial time learnability as well.

Throughout this paper we focus on polynomial query learnability and only provide a brief discussion of our polynomial time learnability results. A more detailed discussion of polynomial time learnability as well as other proof details are provided in an appendix of this paper, available from http://cgi.csc.liv.ac.uk/~frank/publ/publ.html

Related Work Apart from Angluin's classical learning algorithm for propositional Horn, we highlight investigations of exact learnability of fragments of FO Horn (Reddy and Tadepalli 1999; Arias and Khardon 2002; Arias, Khardon, and Maloberti 2007; Selman and Fern 2011) and, more recently, schema mappings (ten Cate, Dalmau, and Kolaitis 2012). $\mathcal{ELH}_{\mathsf{lhs}}$ can be seen as a fragment of FO Horn which, in contrast to many existing approaches, allows recursion and does not impose bounds on the number of variables per clause. In DL, exact learning has been studied for the description logic CLASSIC in (Frazier and Pitt 1996; Cohen and Hirsh 1994) where it is shown that CLASSIC concept expressions (but not TBoxes) can be learned in polynomial time. In this case, membership queries ask whether the target concept subsumes a given concept. Related work on machine learning in DL also include learning DL concept expressions using refinement operators (Lehmann and Hitzler 2010) and completing knowledge bases using formal concept analysis (Baader et al. 2007).

All exact learning frameworks for logical theories considered so far are based on interpretations (Angluin, Frazier, and Pitt 1992; ten Cate, Dalmau, and Kolaitis 2012; Klarman and Britz 2015) or entailments (Frazier and Pitt 1993; Reddy and Tadepalli 1998; Arias and Khardon 2002). In this paper we introduce a new class of examples based on certain answers to data retrieval queries.

Preliminaries

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite sets of *concept* and *role* names, respectively. We begin by introducing members of the $\mathcal{EL}$ family of DLs (Baader, Brandt, and Lutz 2005). An $\mathcal{EL}$ *concept expression* is formed according to the rule $C, D := A \mid \top \mid C \sqcap D \mid \exists r.C$, where A ranges over $\mathsf{N_C}$ and r ranges over $\mathsf{N_R}$. An $\mathcal{EL}$ *concept inclusion (CI)* takes the form $C \sqsubseteq D$, where C and D are $\mathcal{EL}$ concept expressions. An $\mathcal{EL}$ *TBox* $\mathcal{T}$ is a finite set of $\mathcal{EL}$ CIs. An $\mathcal{EL}$ *role inclusion (RI)* takes the form $r \sqsubseteq s$, where $r, s \in \mathsf{N_R}$ and an $\mathcal{EL}$ *RBox* $\mathcal{R}$ is a finite set of $\mathcal{EL}$ role inclusions. The union of an $\mathcal{EL}$ TBox and an $\mathcal{EL}$ RBox is called a $\mathcal{ELH}$ TBox. We also consider the fragments $\mathcal{EL}_{\mathsf{lhs}}$ and $\mathcal{EL}_{\mathsf{rhs}}$ of $\mathcal{EL}$ in which concepts on the right-hand side and, respectively, left-hand side of CIs must be concept names. Thus, $\exists r.A \sqsubseteq B$ is an $\mathcal{EL}_{\mathsf{lhs}}$ CI but not an $\mathcal{EL}_{\mathsf{rhs}}$ CI and $A \sqsubseteq \exists r.B$ is an $\mathcal{EL}_{\mathsf{rhs}}$ CI but not an $\mathcal{EL}_{\mathsf{lhs}}$ CI. By $\mathcal{ELH}_{\mathsf{lhs}}$ and $\mathcal{ELH}_{\mathsf{rhs}}$ we denote the extension of these fragments with $\mathcal{EL}$ RIs.

A *role* is a role name or an inverse role r^- with $r \in \mathsf{N_R}$. The language DL-Lite$^{\exists}_{\mathcal{H}}$ is obtained from $\mathcal{ELH}_{\mathsf{rhs}}$ by admitting both role names and inverse roles in concept expressions and in role inclusions and by admitting, in addition to concept names, *basic concepts* $\exists r.\top$, with r a role, on the left-hand side of CIs. Call an $\mathcal{EL}$ concept expression using inverse roles an $\mathcal{ELI}$ *concept expression*. Then DL-Lite$^{\exists}_{\mathcal{H}}$ coincides with the extension of the language DL-Lite$_{\mathcal{R}}$ (without disjointness constraints) introduced in (Calvanese et al. 2007) with arbitrary $\mathcal{ELI}$ concept expressions on the right-hand side of CIs. The *signature* $\Sigma_{\mathcal{T}}$ of a TBox $\mathcal{T}$ is the set of concept and role names that occur in $\mathcal{T}$.

In description logic, data are stored in ABoxes. Let $\mathsf{N_I}$ be a countably infinite set of *individual names*. An *ABox* $\mathcal{A}$ is a finite non-empty set containing assertions $A(a)$ and $r(a, b)$, where a, b are individuals in $\mathsf{N_I}$, A is a concept name and r is a role. $\mathsf{Ind}(\mathcal{A})$ denotes the set of individuals that occur in $\mathcal{A}$. $\mathcal{A}$ is a *singleton* ABox if it contains only one ABox assertion.

We consider the main query languages for retrieving data from ABoxes using DL TBoxes. An *atomic query (AQ)* q takes the form $A(a)$ or $r(a, b)$, where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. An $\mathcal{EL}$*-instance query (*$\mathcal{EL}$*-IQ)* q takes the form $C(a)$ or $r(a, b)$, where C is an $\mathcal{EL}$ concept expression, $r \in \mathsf{N_R}$ and $a, b \in \mathsf{N_I}$. $\mathcal{ELI}$*-instance queries (*$\mathcal{ELI}$*-IQs)* are defined in the same way by replacing $\mathcal{EL}$ concept expressions with $\mathcal{ELI}$ concept expressions. Finally, a *conjunctive query (CQ)* q is a first-order sentence $\exists \vec{x} \varphi(\vec{a}, \vec{x})$, where φ is a conjunction of atoms of the form $r(t_1, t_2)$ or $A(t)$, where t_1, t_1, t can be individual names from $\vec{a}$ or individual variables from $\vec{x}$. We often slightly abuse notation and denote by AQ the set of AQs and similarly for $\mathcal{EL}$-IQs, $\mathcal{ELI}$-IQs and CQs.

The *size* of a concept expression C (TBox $\mathcal{T}$, ABox $\mathcal{A}$, query q), denoted by $|C|$ (and, respectively, $|\mathcal{T}|$, $|\mathcal{A}|$, and $|q|$) is the length of the word that represents it.

The semantics of DLs is defined as usual (Baader et al. 2003). For an interpretation $\mathcal{I}$, we write $\mathcal{I} \models \alpha$ to state that a CI, RI, ABox assertion, or query α is true in $\mathcal{I}$. An interpretation $\mathcal{I}$ is a *model* of a knowledge base (KB) $(\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T} \cup \mathcal{A}$. We set $(\mathcal{T}, \mathcal{A}) \models \alpha$ and say that α is *entailed* by $(\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \alpha$ for all models $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$.

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 *learning concepts*, and μ is a mapping from $\mathcal{L}$ to 2^X. Given a DL L, the *subsumption* learning framework $\mathfrak{F}_{\mathcal{S}}(L)$, studied in (Konev et al. 2014), is defined as $(X, \mathcal{L}, \mu)$, where $\mathcal{L}$ is the set of all TBoxes that are formulated in L; X is the set of concept and role inclusions α that can occur in TBoxes of L; and $\mu(\mathcal{T})$ is defined as $\{\alpha \in X \mid \mathcal{T} \models \alpha\}$, for every $\mathcal{T} \in \mathcal{L}$. It should be clear that $\mu(\mathcal{T}) = \mu(\mathcal{T}')$ iff the TBoxes $\mathcal{T}$ and $\mathcal{T}'$ entail the same set of inclusions, that is, they are logically equivalent.

For a DL L and query language Q, we study the *data re-*

trieval learning framework $\mathfrak{F}_{\mathcal{D}}(L,Q)$ defined as $(X,\mathcal{L},\mu)$, where $\mathcal{L}$ is again the set of all TBoxes that are formulated in L; X is the set of *data retrieval examples* of the form $(\mathcal{A},q)$, where $\mathcal{A}$ is an ABox and $q\in Q$; and $\mu(\mathcal{T})=\{(\mathcal{A},q)\in X\mid(\mathcal{T},\mathcal{A})\models q\}$. We only consider data retrieval frameworks $\mathfrak{F}_{\mathcal{D}}(L,Q)$ in which $\mu(\mathcal{T})=\mu(\mathcal{T}')$ iff the TBoxes $\mathcal{T}$ and $\mathcal{T}'$ are logically equivalent. Note that this is not the case for the pairs (L,AQ) with L from $\mathcal{EL}_{\mathsf{rhs}}(\mathcal{H})$, $\mathcal{EL}(\mathcal{H})$, DL-Lite$^{\exists}_{\mathcal{H}}$, and for the pair (DL-Lite$^{\exists}_{\mathcal{H}}$, $\mathcal{EL}$-IQ) (see Table 1). For example, for the $\mathcal{EL}$ TBoxes $\mathcal{T}_1=\{A\sqsubseteq\exists r.\top\}$ and $\mathcal{T}_2=\{A\sqsubseteq\exists r.\exists r.\top\}$ we have $(\mathcal{T}_1,\mathcal{A})\models q$ iff $(\mathcal{T}_2,\mathcal{A})\models q$ for every ABox $\mathcal{A}$ and AQ q. Thus, $\mathcal{T}_1$ and $\mathcal{T}_2$ cannot be distinguished using data retrieval examples based on AQs and so $\mathcal{EL}$ TBoxes cannot be learned using such examples.

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

We say that a learning framework $(X,\mathcal{L},\mu)$ is *exact learnable* if there is an algorithm A such that for any target $l\in\mathcal{L}$ the algorithm A always halts and outputs $l'\in\mathcal{L}$ such that $\mu(l)=\mu(l')$ using membership and equivalence queries answered by the oracles $\mathsf{MEM}_{l,X}$ and $\mathsf{EQ}_{l,X}$, respectively. at any stage in a run A learning framework $(X,\mathcal{L},\mu)$ is *polynomial query* exact 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).

An important class of learning algorithms—in particular, all algorithms presented in (Konev et al. 2014; Frazier and Pitt 1993; Reddy and Tadepalli 1998) fit in this class—is the algorithm in which the hypothesis h of any equivalence query is of polynomial size in l and such that $\mu(h)\subseteq\mu(l)$. Then counterexamples returned by the $\mathsf{EQ}_{l,X}$ oracles are always positive. We say that such algorithms use *positive bounded equivalence queries*. The learning algorithms studied in this paper are positive and, therefore, the equivalence queries posed to the domain expert are in fact completeness queries that ask whether the hypothesis entails the target TBox.

Polynomial Query Learnability

In this section we prove the positive results presented in Table 1 for the data retrieval setting by reduction to the subsumption setting. We employ the following result based on (Konev et al. 2014), except for $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{lhs}})$ which is proved in the appendix by extending the proof for $\mathfrak{F}_{\mathcal{S}}(\mathcal{EL}_{\mathsf{lhs}})$ in (Konev et al. 2014).

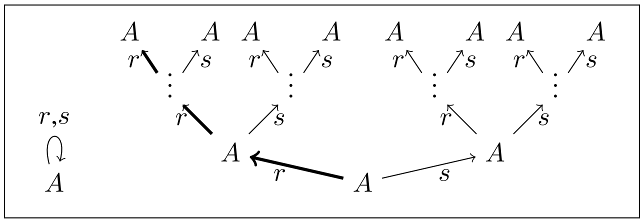

Figure 1: An ABox $\mathcal{A}=\{r(a,a),s(a,a),A(a)\}$ and its unravelling up to level n.

Theorem 1 *The subsumption learning frameworks $\mathfrak{F}_{\mathcal{S}}(\mathcal{EL}_{\mathsf{lhs}})$, $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{lhs}})$, $\mathfrak{F}_{\mathcal{S}}(\mathcal{EL}_{\mathsf{rhs}})$, $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{rhs}})$ and $\mathfrak{F}_{\mathcal{S}}$(DL-Lite$^{\exists}_{\mathcal{H}}$) are polynomial query exact learnable with membership and positive bounded equivalence queries.*

We begin by illustrating the idea of the reduction for $\mathcal{EL}_{\mathsf{lhs}}$ and AQs. To learn a TBox from data retrieval examples we run a learning from subsumptions algorithm as a 'black box'. Every time the learning from subsumptions algorithm makes a membership or an equivalence query we rewrite the query into the data setting and pass it on to the data retrieval oracle. The oracle's answer, rewritten back to the subsumption setting, is given to the learning from subsumptions algorithm. When the learning from subsumptions algorithm terminates we return the learnt TBox. This reduction is made possible by the close relationship between data retrieval and subsumption examples. For every TBox $\mathcal{T}$ and inclusions $C\sqsubseteq B$, one can interpret a concept expression C as a labelled tree and encode this tree as an ABox $\mathcal{A}_C$ with root ρ_C such that $\mathcal{T}\models C\sqsubseteq B$ iff $(\mathcal{T},\mathcal{A}_C)\models B(\rho_C)$.

Then, membership queries in the subsumption setting can be answered with the help of a data retrieval oracle due to the relation between subsumptions and AQs described above. An inclusion $C\sqsubseteq B$ is a (positive) subsumption example for some target TBox $\mathcal{T}$ if, and only if, $(\mathcal{A}_C,B(\rho_C))$ is a (positive) data retrieval example for the same target $\mathcal{T}$. To handle equivalence queries, we need to be able to rewrite data retrieval counterexamples returned by the data retrieval oracle into the subsumption setting. For every TBox $\mathcal{T}$ and data retrieval query $(\mathcal{A},B(a))$ one can construct a concept expression $C_{\mathcal{A}}$ such that $(\mathcal{T},\mathcal{A})\models B(a)$ iff $\mathcal{T}\models C_{\mathcal{A}}\sqsubseteq B$. Such a concept expression $C_{\mathcal{A}}$ can be obtained by unravelling $\mathcal{A}$ into a tree-shaped ABox and representing it as a concept expression. This unravelling, however, can increase the ABox size exponentially. Thus, to obtain a polynomial query bound on the the learning process, $C_{\mathcal{A}}\sqsubseteq D$ cannot be simply returned as an answer to a subsumption equivalence query.

For example, for a target TBox $\mathcal{T}=\{\exists r^n.A\sqsubseteq B\}$ and a hypothesis $\mathcal{H}=\emptyset$ the data retrieval query $(\mathcal{A},B(a))$, where $\mathcal{A}=\{r(a,a),s(a,a),A(a)\}$, is a positive counterexample. The tree-shaped unravelling of $\mathcal{A}$ up to level n is a full binary tree of depth n, as shown in Fig. 1. On the other hand, the non-equivalence of $\mathcal{T}$ and $\mathcal{H}$ can already be witnessed by $(\mathcal{A}',B(a))$, where $\mathcal{A}'=\{r(a,a),A(a)\}$. The unravelling of $\mathcal{A}'$ up to level

n produces a linear size ABox $\{r(a,a_2), r(a_2,a_3), \ldots, r(a_{n-1},a_n), A(a), A(a_2), \ldots, A(a_n)\}$, corresponding to the left-most path in Fig. 1, which, in turn, is linear-size w.r.t. the target inclusion $\exists r^n.A \sqsubseteq B$. Notice that $\mathcal{A}'$ is obtained from $\mathcal{A}$ by removing the $s(a,a)$ edge and checking, using membership queries, whether $(\mathcal{T},\mathcal{A}') \models q$ still holds. In other words, one might need to ask further membership queries in order to rewrite answers to data retrieval equivalence queries given by the data retrieval oracle into the subsumption setting.

We address the need of rewriting counterexamples by introducing an abstract notion of reduction between different exact learning frameworks. To simplify, we assume that both learning frameworks use the same set of learning concepts $\mathcal{L}$ and only consider positive bounded equivalence queries. We say that a learning framework $\mathfrak{F} = (X,\mathcal{L},\mu)$ *positively polynomial query reduces* to $\mathfrak{F}' = (X',\mathcal{L},\mu')$ if, for any $l,h \in \mathcal{L}$, $\mu(h) \subseteq \mu(l)$ if, and only if, $\mu'(h) \subseteq \mu'(l)$; and for some polynomials $p_1(\cdot)$, $p_2(\cdot)$ and $p_3(\cdot,\cdot)$ there exist a function $f_{\mathsf{MEM}} : X' \to X$, translating an $\mathfrak{F}'$ membership query to $\mathfrak{F}$, and a partial function $f_{\mathsf{EQ}} : \mathcal{L}\times\mathcal{L}\times X \to X'$ defined for every (l,h,x) such that $|h| \leq p_1(|l|)$, translating an answer to an $\mathfrak{F}$ equivalence query to $\mathfrak{F}'$, such that:

- for all $x' \in X'$ we have $x' \in \mu'(l)$ iff $f_{\mathsf{MEM}}(x') \in \mu(l)$;
- for all $x \in X$ we have $x \in \mu(l) \setminus \mu(h)$ iff $f_{\mathsf{EQ}}(l,h,x) \in \mu'(l) \setminus \mu'(h)$;
- $|f_{\mathsf{MEM}}(x')| \leq p_2(|x'|)$;
- the sum of sizes of inputs to queries used to compute $f_{\mathsf{EQ}}(l,h,x)$ is bounded by $p_3(|l|,|x|)$, $|f_{\mathsf{EQ}}(l,h,x)| \leq p_3(|l|,|x|)$ and l can only be accessed by calls to the oracle $\mathsf{MEM}_{l,X}$.

Note that even though f_{EQ} takes h as input, the polynomial query bound on computing $f_{\mathsf{EQ}}(l,h,x)$ does not depend on the size of h as f_{EQ} is only defined for h polynomial in the size of l.

Theorem 2 *Let $\mathfrak{F} = (X,\mathcal{L},\mu)$ and $\mathfrak{F}' = (X',\mathcal{L},\mu')$ be learning frameworks. If there exists a positive polynomial query reduction from $\mathfrak{F}$ to $\mathfrak{F}'$ and a polynomial query learning algorithm for $\mathfrak{F}'$ that uses membership queries and positive bounded equivalence queries then $\mathfrak{F}$ is polynomial query exact learnable.*

We use Theorem 2 to prove polynomial query learnability of $\mathfrak{F}_{\mathcal{D}}$(DL-Lite$^{\exists}_{\mathcal{H}}$, $\mathcal{ELI}$-IQ) and $\mathfrak{F}_{\mathcal{D}}(\mathcal{ELH}_{\mathsf{lhs}}$, AQ) by reduction to $\mathfrak{F}_{\mathcal{S}}$(DL-Lite$^{\exists}_{\mathcal{H}}$) and, respectively, $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{lhs}})$. The remaining positive results in Table 1 are similar and given in the appendix.

The function f_{MEM} required in Theorem 2 is easily defined by setting $f_{\mathsf{MEM}}(r \sqsubseteq s) := (\{r(a,b)\}, s(a,b))$ (for distinct $a,b \in \mathsf{N_I}$) and $f_{\mathsf{MEM}}(C \sqsubseteq D) := (\mathcal{A}_C, D(\rho_C))$ since $(\mathcal{T},\{r(a,b)\}) \models s(a,b)$ iff $\mathcal{T} \models r \sqsubseteq s$ and $(\mathcal{T},\mathcal{A}_C) \models D(\rho_C)$ iff $\mathcal{T} \models C \sqsubseteq D$.

Conversely, given a positive counterexample $(\mathcal{A}, r(a,b))$ (that is, $(\mathcal{T},\mathcal{A}) \models r(a,b)$ and $(\mathcal{H},\mathcal{A}) \not\models r(a,b)$ for target TBox $\mathcal{T}$ and hypothesis $\mathcal{H}$) there always exists $s(a,b) \in \mathcal{A}$ such that $(\{s(a,b)\}, r(a,b))$ is a positive counterexample as well. Thus, we define $f_{\mathsf{EQ}}(\mathcal{T},\mathcal{H},(\mathcal{A},r(a,b))) := s \sqsubseteq r$. In what follows we define the image of f_{EQ} for counterexamples of the form $(\mathcal{A}, C(a))$.

Algorithm 1 Reducing a positive counterexample

```
 1: function ReduceCounterExample( $\mathcal{A}$, $C(a)$ )
 2:     Find a role saturated and parent/child merged $C(a)$
 3:     if $C = C_0 \sqcap ... \sqcap C_n$ then
 4:         Find $C_i$, $0 \leq i \leq n$, such that $(\mathcal{H},\mathcal{A}) \not\models C_i(a)$
 5:         $C := C_i$
 6:     if $C = \exists r.C'$ and there is $s(a,b) \in \mathcal{A}$ such that
 7:      $(\mathcal{T},\{s(a,b)\}) \models r(a,b)$ and $(\mathcal{T},\mathcal{A}) \models C'(b)$ then
 8:         ReduceCounterExample( $\mathcal{A}$, $C'(b)$ )
 9:     else
10:         Find a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that
11:                 $(\mathcal{T},\mathcal{A}') \models C(a)$ but $(\mathcal{H},\mathcal{A}') \not\models C(a)$
12:         return ($\mathcal{A}'$,$C(a)$)
```

Construction of f_{EQ} for $\mathfrak{F}_{\mathcal{D}}$(DL-Lite$^{\exists}_{\mathcal{H}}$, $\mathcal{ELI}$-IQ) Given a target $\mathcal{T}$ and hypothesis $\mathcal{H}$ such that $\mathcal{T} \models \mathcal{H}$, Algorithm 1 transforms every positive counterexample $(\mathcal{A}, C(a))$ into a positive counterexample $(\mathcal{A}', D(b))$ where $\mathcal{A}' \subseteq \mathcal{A}$ is a singleton ABox (i.e., of the form $\{A(a)\}$ or $\{r(a,b)\}$). Using the equivalences $(\mathcal{T},\{A(b)\}) \models D(b)$ iff $\mathcal{T} \models A \sqsubseteq D$ and $(\mathcal{T},\{r(b,c)\}) \models D(b)$ iff $\mathcal{T} \models \exists r.\top \sqsubseteq D$, we then obtain a positive subsumption counterexample which will be the image of $(\mathcal{T},\mathcal{H},(\mathcal{A},C(a)))$ under f_{EQ}.

Given a positive data retrieval counterexample $(\mathcal{A}, C(a))$, Algorithm 1 exhaustively applies the *role saturation* and *parent-child merging* rules introduced in (Konev et al. 2014). We say that an $\mathcal{ELI}$-IQ $C(a)$ is *role saturated* for $(\mathcal{T},\mathcal{A})$ if $(\mathcal{T},\mathcal{A}) \not\models C'(a)$ whenever C' is the result of replacing an occurrence of a role r by some role s with $\mathcal{T} \not\models r \sqsubseteq s$ and $\mathcal{T} \models s \sqsubseteq r$. To define parent/child merging, we identify each $\mathcal{ELI}$ concept C with a finite tree T_C whose nodes are labeled with concept names and edges are labeled with roles. For example, if $C = \exists t.(A \sqcap \exists r.\exists r^-.\exists r.B) \sqcap \exists s.\top$ then Fig. 2a illustrates T_C. Now, we say that an $\mathcal{ELI}$-IQ $C(a)$ is *parent/child merged* for $\mathcal{T}$ and $\mathcal{A}$ if for nodes n_1, n_2, n_3 in T_C such that n_2 is an r-successor of n_1, n_3 is an s-successor of n_2 and $\mathcal{T} \models r^- \equiv s$ we have $(\mathcal{T},\mathcal{A}) \not\models C'(a)$ if C' is the concept that results from identifying n_1 and n_3. For instance, the concept in Fig. 2c is the result of identifying the leaf labeled with B in Fig. 2b with the parent of its parent. The corresponding role saturation and parent-child merging rules are formulated in the obvious way.

In Algorithm 1 the learner uses membership queries in Lines 2, 7 and 10-11. We present a run for $\mathcal{T} = \{A \sqsubseteq \exists s.B, s \sqsubseteq r\}$ and $\mathcal{H} = \{s \sqsubseteq r\}$. Assume the oracle gives as counterexample $(\mathcal{A}, C(a))$, where $\mathcal{A} = \{t(a,b), A(b), s(a,c)\}$ and $C(a) = \exists t.(A \sqcap \exists r.\exists r^-.\exists r.B) \sqcap \exists s.\top(a)$ (Fig. 2a). Role saturation produces $C(a) = \exists t.(A \sqcap \exists s.\exists s^-.\exists s.B) \sqcap \exists s.\top(a)$ (Fig. 2b). Then, applying parent/child merging twice we obtain $C(a) = \exists t.(A \sqcap \exists s.B) \sqcap \exists s.\top(a)$ (Fig. 2c and 2d).

Since $(\mathcal{H},\mathcal{A}) \not\models \exists t.(A \sqcap \exists s.B)(a)$, after Lines 3-5, Algorithm 1 updates C by choosing the conjunct $\exists t.(A \sqcap \exists s.B)$. As C is of the form $\exists t.C'$ and there is $t(a,b) \in \mathcal{A}$ such

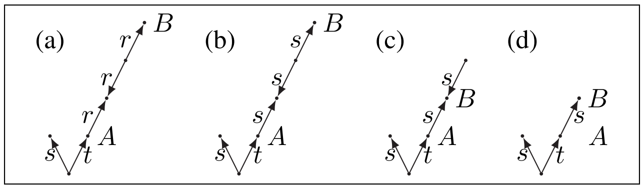

Figure 2: Concept C being role saturated and parent/child merged.

that $(\mathcal{T},\mathcal{A}) \models C'(b)$, the algorithm recursively calls the function "ReduceCounterExample" with $A \sqcap \exists s.B(b)$. Now, since $(\mathcal{H},\mathcal{A}) \not\models \exists s.B(b)$, after Lines 3-5, C is updated to $\exists s.B$. Finally, C is of the form $\exists t.C'$ and there is no $t(b,c) \in \mathcal{A}$ such that $(\mathcal{T},\mathcal{A}) \models C'(c)$. So the algorithm proceeds to Lines 10-11, where it chooses $A(b) \in \mathcal{A}$. Since $(\mathcal{T},\{A(b)\}) \models \exists s.B(b)$ and $(\mathcal{H},\{A(b)\}) \not\models \exists s.B(b)$ we have that $\mathcal{T} \models A \sqsubseteq \exists s.B$ and $\mathcal{H} \not\models A \sqsubseteq \exists s.B$.

The following two lemmas state the main properties of Algorithm 1. A detailed analysis is given in the appendix.

Lemma 3 *Let $(\mathcal{A}, C(a))$ be a positive counterexample. Then the following holds:*

1. *if C is a basic concept then there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T},\mathcal{A}') \models C(a)$;*
2. *if C is of the form $\exists r.C'$ and C is role saturated and parent/child merged then either there is $s(a,b) \in \mathcal{A}$ (where r, s are roles) such that $(\mathcal{T},\{s(a,b)\}) \models r(a,b)$ and $(\mathcal{T},\mathcal{A}) \models C'(b)$ or there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T},\mathcal{A}') \models C(a)$.*

Lemma 4 *For any DL-Lite$^{\exists}_{\mathcal{H}}$ target $\mathcal{T}$ and any DL-Lite$^{\exists}_{\mathcal{H}}$ hypothesis $\mathcal{H}$ with size polynomial in $|\mathcal{T}|$, given a positive counterexample $(\mathcal{A}, C(a))$, Algorithm 1 computes with polynomially many polynomial size queries in $|\mathcal{T}|$, $|\mathcal{A}|$ and $|C|$ a positive counterexample $(\mathcal{A}', D(b))$, where $\mathcal{A}' \subseteq \mathcal{A}$ is a singleton ABox.*

Proof. (Sketch) Let $(\mathcal{A}, C(a))$ be the input of "ReduceCounterExample". The computation of Line 2 requires polynomially many polynomial size queries in $|C|$ and $|\mathcal{T}|$. If C has more than one conjunct then it is updated in Lines 3-5, so C becomes either (1) a basic concept or (2) of the form $\exists r.C'$. By Lemma 3 in case (1) there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T},\mathcal{A}') \models C(a)$, computed by Lines 10-11 of Algorithm 1. In case (2) either there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T},\mathcal{A}') \models C(a)$, computed by Lines 10-11 of Algorithm 1, or we obtain a counterexample with a refined C. Since the size of the refined counterexample is strictly smaller after every recursive call of "ReduceCounterExample", the total number of calls is bounded by $|C|$. ❑

Using Theorem 1 and Theorem 2 we now obtain that $\mathfrak{F}_{\mathcal{D}}$(DL-Lite$^{\exists}_{\mathcal{H}}$, $\mathcal{ELI}$-IQ) is polynomial query exact learnable.

Construction of f_{EQ} for $\mathfrak{F}_{\mathcal{D}}(\mathcal{ELH}_{\mathsf{lhs}}, \mathbf{AQ})$ We first transform a positive counterexample of the form $(\mathcal{A}, A(a))$ into a positive counterexample of the form $(\mathcal{A}', B(\rho_{\mathcal{A}'}))$ with $\mathcal{A}'$ a tree-shaped ABox rooted in $\rho_{\mathcal{A}'}$. We then define the image of $(\mathcal{T},\mathcal{H},(\mathcal{A},A(a)))$ under f_{EQ} as $C_{\mathcal{A}'} \sqsubseteq B$, where $C_{\mathcal{A}'}$ is the $\mathcal{EL}$ concept expression corresponding to $\mathcal{A}'$. Our algorithm is based on two operations: *minimization*, computed by Algorithm 2, and *cycle unfolding*. Algorithm 2 *minimizes* a given ABox with the following three rules:

Algorithm 2 Minimizing an ABox $\mathcal{A}$

```
1: function MINIMIZEABOX( $\mathcal{A}$)
2:     Concept saturate $\mathcal{A}$ with $\mathcal{H}$
3:     for every $A \in \mathsf{N_C} \cap \Sigma_{\mathcal{T}}$ and $a \in \mathsf{Ind}(\mathcal{A})$ such that
4:         $(\mathcal{T},\mathcal{A}) \models A(a)$ and $(\mathcal{H},\mathcal{A}) \not\models A(a)$ do
5:         Domain Minimize $\mathcal{A}$ with $A(a)$
6:         Role Minimize $\mathcal{A}$ with $A(a)$
7:     return ($\mathcal{A}$)
```

(Concept saturate $\mathcal{A}$ with $\mathcal{H}$) If $A(a) \notin \mathcal{A}$ and $(\mathcal{H},\mathcal{A}) \models A(a)$ then replace $\mathcal{A}$ by $\mathcal{A} \cup \{A(a)\}$, where $A \in \mathsf{N_C} \cap \Sigma_{\mathcal{T}}$ and $a \in \mathsf{Ind}(\mathcal{A})$.

(Domain Minimize $\mathcal{A}$ with $A(a)$) If $(\mathcal{A}, A(a))$ is a counterexample and $(\mathcal{T}, \mathcal{A}^{-b}) \models A(a)$ then replace $\mathcal{A}$ by $\mathcal{A}^{-b}$, where $\mathcal{A}^{-b}$ is the result of removing from $\mathcal{A}$ all ABox assertions in which b occurs.

(Role Minimize $\mathcal{A}$ with $A(a)$) If $(\mathcal{A}, A(a))$ is a counterexample and $(\mathcal{T}, \mathcal{A}^{-r(b,c)}) \models A(a)$ then replace $\mathcal{A}$ by $\mathcal{A}^{-r(b,c)}$, where $\mathcal{A}^{-r(b,c)}$ is obtained by removing a role assertion $r(b,c)$ from $\mathcal{A}$.

Lemma 5 *For any $\mathcal{ELH}_{\mathsf{lhs}}$ target $\mathcal{T}$ and any $\mathcal{ELH}_{\mathsf{lhs}}$ hypothesis $\mathcal{H}$ with size polynomial in $|\mathcal{T}|$, given a positive counterexample $(\mathcal{A}, A(a))$, Algorithm 2 computes, with polynomially many polynomial size queries in $|\mathcal{A}|$ and $|\mathcal{T}|$, an ABox $\mathcal{A}'$ such that $|\mathcal{A}'| \leq |\mathcal{T}|$ and there exists an AQ $A'(a')$ such that $(\mathcal{A}', A'(a'))$ is a positive counterexample.*

It remains to show that the ABox can be made tree-shaped. We say that an ABox $\mathcal{A}$ has an (undirected) cycle if there is a finite sequence $a_0 \cdot r_1 \cdot a_1 \cdot ... \cdot r_k \cdot a_k$ such that (i) $a_0 = a_k$ and (ii) there are mutually distinct assertions of the form $r_{i+1}(a_i, a_{i+1})$ or $r_{i+1}(a_{i+1}, a_i)$ in $\mathcal{A}$, for $0 \leq i < k$. The *unfolding* of a cycle $c = a_0 \cdot r_1 \cdot a_1 \cdot ... \cdot r_k \cdot a_k$ in a given ABox $\mathcal{A}$ is obtained by replacing c by the cycle $c' = a_0 \cdot r_1 \cdot a_1 \cdot ... \cdot r_k \cdot a_{k-1} \cdot r_k \cdot \widehat{a}_0 \cdot r_1 \cdots \widehat{a}_{k-1} \cdot r_k \cdot a_0$, where $\widehat{a}_i$ are fresh individual names, $0 \leq i \leq k-1$, in such a way that (i) if $r(a_i, d) \in \mathcal{A}$, for an individual d not in the cycle, then $r(\widehat{a}_i, d) \in \mathcal{A}$; and (ii) if $A(a_i) \in \mathcal{A}$ then $A(\widehat{a}_i) \in \mathcal{A}$.

We prove in the appendix that after every cycle unfolding/minimisation step in Algorithm 3 the ABox $\mathcal{A}$ on the one hand becomes strictly larger and on the other does not exceed the size of the target TBox $\mathcal{T}$. Thus Algorithm 3 terminates after a polynomial number of steps yielding a tree-shaped (by Line 3) ABox $\mathcal{A}$ such that $(\mathcal{A}, B(\rho_{\mathcal{A}}))$ is a positive counterexample.

Lemma 6 *For any $\mathcal{ELH}_{\mathsf{lhs}}$ target $\mathcal{T}$ and any $\mathcal{ELH}_{\mathsf{lhs}}$ hypothesis $\mathcal{H}$ with size polynomial in $|\mathcal{T}|$, given a positive counterexample $(\mathcal{A}, A(a))$, Algorithm 3 computes, with polynomially many polynomial size queries in $|\mathcal{T}|$ and $|\mathcal{A}|$, a tree shaped ABox $\mathcal{A}$ rooted in $\rho_{\mathcal{A}}$ and $B \in \mathsf{N_C}$ such that $(\mathcal{A}, B(\rho_{\mathcal{A}}))$ is a positive counterexample.*

Algorithm 3 Computing a tree shaped ABox

```
1: function FINDTREE($\mathcal{A}$)
2:     MINIMIZEABOX($\mathcal{A}$)
3:     while there is a cycle $c$ in $\mathcal{A}$ do
4:         Unfold $a \in \mathsf{Ind}(\mathcal{A})$ in cycle $c$
5:         MINIMIZEABOX($\mathcal{A}$)
6:     Find $B \in \mathsf{N_C} \cap \Sigma_{\mathcal{T}}$ such that for root $\rho_{\mathcal{A}}$ of $\mathcal{A}$
7:             $(\mathcal{T}, \mathcal{A}) \models B(\rho_{\mathcal{A}})$ but $(\mathcal{H}, \mathcal{A}) \not\models B(\rho_{\mathcal{A}})$
8:     return $(\mathcal{A}, B(\rho_{\mathcal{A}}))$
```

Using Theorem 1 and Theorem 2 we obtain that the learning framework $\mathfrak{F}_{\mathcal{D}}(\mathcal{ELH}_{\mathsf{lhs}}, \text{AQ})$ is polynomial query exact learnable.

Limits of Polynomial Query Learnability

We prove that $\mathcal{EL}_{\mathsf{rhs}}$ TBoxes are not polynomial query learnable using data retrieval examples with CQs. This is in contrast to polynomial query learnability of DL-Lite$^{\exists}_{\mathcal{H}}$ and $\mathcal{EL}_{\mathsf{rhs}}$ TBoxes using data retrieval examples with $\mathcal{ELI}$-IQs and, respectively, $\mathcal{EL}$-IQs. Surprisingly, the negative result holds already if queries of the form $\exists x.A(x)$ are admitted in addition to $\mathcal{EL}$-IQs.

To prove our result, we define a superpolynomial set S of TBoxes and show that (i) any polynomial size membership query can distinguish at most polynomially many TBoxes from S; and (ii) there exist superpolynomially many polynomial size data retrieval examples that the oracle can give as counterexamples which distinguish at most one TBox from S. To present the TBoxes in S, fix two role names r and s. For any sequence $\boldsymbol{\sigma} = \sigma^1\sigma^2\ldots\sigma^n$ with $\sigma^i \in \{r, s\}$, the expression $\exists\boldsymbol{\sigma}.C$ stands for $\exists\sigma^1.\exists\sigma^2\ldots\exists\sigma^n.C$. Denote by $\mathfrak{L}$ the set of all sequences $\boldsymbol{\sigma}$, of which there are $N = 2^n$ many. For every such sequence $\boldsymbol{\sigma}$, consider the $\mathcal{EL}_{\mathsf{rhs}}$ TBox $\mathcal{T}_{\boldsymbol{\sigma}}$ defined as

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

Here the X_i are used to generate a binary tree of depth n from the ABox $\{A(a)\}$. The inclusion $A \sqsubseteq \exists\boldsymbol{\sigma}.M$ singles out one path in this tree for each $\mathcal{T}_{\boldsymbol{\sigma}}$. Finally, whenever M holds, then each $\mathcal{T}_{\boldsymbol{\sigma}}$ generates an infinite binary tree with Ms. Denote by $\Gamma_n = \{r, s, A, M, X_0, \ldots, X_n\}$ the signature of the TBoxes $\mathcal{T}_{\boldsymbol{\sigma}} \in S$. Notice that $\mathcal{T}_0$ is easy to learn. Moreover, if $\mathcal{T}_0$ is known to the learner and only IQs are available in responses to equivalences queries, then a single equivalence query can force the oracle to reveal $\mathcal{T}_{\boldsymbol{\sigma}}$ as $A \sqsubseteq \exists\boldsymbol{\sigma}.M$ can be found 'inside' every counterexample. On the other hand, if CQs are used then the oracle can provide counterexamples of the form $(\{A(a)\}, \exists x.M(x))$, without giving any useful information about $\mathcal{T}_{\boldsymbol{\sigma}}$. Points (i) and (ii) above follow from Lemma 7 and, respectively, Lemma 8, proved in the appendix.

Lemma 7 *For any ABox $\mathcal{A}$ and CQ q over Γ_n either:*

- *for every $\mathcal{T}_{\boldsymbol{\sigma}} \in S$, $(\mathcal{T}_{\boldsymbol{\sigma}}, \mathcal{A}) \models q$; or*
- *the number of $\mathcal{T}_{\boldsymbol{\sigma}} \in S$ such that $(\mathcal{T}_{\boldsymbol{\sigma}}, \mathcal{A}) \models q$ does not exceed $|q|$.*

Lemma 8 *For any $n > 1$ and any $\mathcal{EL}_{\mathsf{rhs}}$ TBox $\mathcal{H}$ over Γ_n there are a singleton ABox $\mathcal{A}$ over Γ_n and a query q that is an $\mathcal{EL}$-IQ over Γ_n with $|q| \leq n+1$ or of the form $q = \exists x.M(x)$ such that either:*

- *$(\mathcal{H}, \mathcal{A}) \models q$ and $(\mathcal{T}_{\boldsymbol{\sigma}}, \mathcal{A}) \models q$ for at most one $\mathcal{T}_{\boldsymbol{\sigma}} \in S$; or*
- *$(\mathcal{H}, \mathcal{A}) \not\models q$ and for every $\mathcal{T}_{\boldsymbol{\sigma}} \in S$ we have $(\mathcal{T}_{\boldsymbol{\sigma}}, \mathcal{A}) \models q$.*

Lemmas 7 and 8 together imply that $\mathcal{EL}_{\mathsf{rhs}}$ TBoxes are not polynomial query learnable usings CQs in data retrieval examples. Moreover, it is sufficient to admit CQs of the form $\exists x.M(x)$ in addition to $\mathcal{EL}$-IQs.

The two lemmas above hold for $\mathcal{ELH}_{\mathsf{rhs}}$, $\mathcal{EL}$, and $\mathcal{ELH}$ as well. This proves the non polynomial query learnability results involving CQs in Table 1. The non polynomial query learnability for $\mathfrak{F}_{\mathcal{D}}(\mathcal{EL}, \mathcal{EL}\text{-IQ})$ and $\mathfrak{F}_{\mathcal{D}}(\mathcal{ELH}, \mathcal{EL}\text{-IQ})$ are proved in the appendix by a nontrivial extension of the non polynomial query learnability result for $\mathcal{EL}$ TBoxes from subsumptions in (Konev et al. 2014).

Polynomial Time Learnability

We briefly comment on our results for polynomial time learnability. The learning algorithm for $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{lhs}})$ in the appendix of this paper and the learning algorithms for $\mathfrak{F}_{\mathcal{S}}(\mathcal{EL}_{\mathsf{lhs}})$, $\mathfrak{F}_{\mathcal{S}}(\mathcal{EL}_{\mathsf{rhs}})$ and $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{rhs}})$ given in (Konev et al. 2014) are in fact polynomial *time* algorithms. Thus, we obtain:

Theorem 9 *The subsumption learning frameworks $\mathfrak{F}_{\mathcal{S}}(\mathcal{EL}_{\mathsf{lhs}})$, $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{lhs}})$, $\mathfrak{F}_{\mathcal{S}}(\mathcal{EL}_{\mathsf{rhs}})$, and $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{rhs}})$ are polynomial time exact learnable with membership and positive bounded equivalence queries.*

Then one can modify the notion of positive polynomial *query* reducibility to an appropriate notion of positive polynomial *time* reducibility and provide positive polynomial time reductions to prove that the results of Table 1 for $\mathcal{EL}_{\mathsf{lhs}}$, $\mathcal{ELH}_{\mathsf{lhs}}$, $\mathcal{EL}_{\mathsf{rhs}}$ and $\mathcal{ELH}_{\mathsf{rhs}}$ hold for polynomial time learnability as well.

Open Problems

A great number of challenging problems remain open. Firstly, it would be of great in interest to find out whether $\mathfrak{F}_{\mathcal{S}}(\text{DL-Lite}^{\exists}_{\mathcal{H}})$ and $\mathfrak{F}_{\mathcal{D}}(\text{DL-Lite}^{\exists}_{\mathcal{H}}, \mathcal{ELI}\text{-IQ})$ are not only polynomial query learnable but also polynomial time learnable. We conjecture that this is not the case (if P≠NP) but did not yet find a way of proving this. Secondly, as stated in Table 1, polynomial query learnability of $\mathfrak{F}_{\mathcal{D}}(\mathcal{EL}, \mathcal{ELI}\text{-IQ})$ and $\mathfrak{F}_{\mathcal{D}}(\text{DL-Lite}^{\exists}_{\mathcal{H}}, \text{CQ})$ remain open problems. Polynomial time learnability of those frameworks is open as well. In both cases we conjecture non polynomial query (and, therefore, time) learnability but a significant modification of the techniques introduced here will be required to prove this. Finally, it would be of interest to apply modified versions of the algorithms presented here to obtain worst-case exponential but practical algorithms for frameworks that are not polynomial query learnable. Examples one might consider are the DLs $\mathcal{EL}$ and $\mathcal{ELH}$ with either subsumption queries or data retrieval queries.

Acknowledgements Ozaki is supported by the Science without Borders scholarship programme.

References

Angluin, D.; Frazier, M.; and Pitt, L. 1992. Learning conjunctions of Horn clauses. *Machine Learning* 9:147–164.

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

Arias, M., and Khardon, R. 2002. Learning closed Horn expressions. *Inf. Comput.* 178(1):214–240.

Arias, M.; Khardon, R.; and Maloberti, J. 2007. Learning Horn expressions with LOGAN-H. *Journal of Machine Learning Research* 8:549–587.

Arias, M. 2004. *Exact learning of first-order expressions from queries*. Ph.D. Dissertation, Citeseer.

Baader, F.; Calvanese, D.; McGuiness, D.; Nardi, D.; and Patel-Schneider, P. 2003. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.

Baader, F.; Ganter, B.; Sertkaya, B.; and Sattler, U. 2007. Completing description logic knowledge bases using formal concept analysis. In *IJCAI*, volume 7, 230–235.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, 364–369. Professional Book Center.

Bechhofer, S.; Horrocks, I.; Goble, C.; and Stevens, R. 2001. Oiled: a reason-able ontology editor for the semantic web. In *KI*. Springer. 396–408.

Bienvenu, M.; Ortiz, M.; Šimkus, M.; and Xiao, G. 2013. Tractable queries for lightweight description logics. In *AAAI*, 768–774. AAAI Press.

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

Buitelaar, P.; Cimiano, P.; and Magnini, B., eds. 2005. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press.

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

Cimiano, P.; Hotho, A.; and Staab, S. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res. (JAIR)* 24:305–339.

Cohen, W. W., and Hirsh, H. 1994. Learning the CLASSIC description logic: Theoretical and experimental results. In *KR*, 121–133. Morgan Kaufmann.

Day-Richter, J.; Harris, M. A.; Haendel, M.; Lewis, S.; et al. 2007. Obo-edit an ontology editor for biologists. *Bioinformatics* 23(16):2198–2200.

Frazier, M., and Pitt, L. 1993. Learning From Entailment: An Application to Propositional Horn Sentences. In *ICML*, 120–127.

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

Kikot, S.; Kontchakov, R.; and Zakharyaschev, M. 2011. On (in) tractability of OBDA with OWL 2 QL. CEUR Workshop Proceedings.

Klarman, S., and Britz, K. 2015. Ontology learning from interpretations in lightweight description logics. In *ILP*.

Konev, B.; Lutz, C.; Ozaki, A.; and Wolter, F. 2014. Exact learning of lightweight description logic ontologies. In *KR*.

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

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

Ma, Y., and Distel, F. 2013. Learning formal definitions for snomed CT from text. In *AIME*, 73–77.

Musen, M. A. 2013. Protégé ontology editor. *Encyclopedia of Systems Biology* 1763–1765.

Reddy, C., and Tadepalli, P. 1998. Learning First-Order Acyclic Horn Programs from Entailment. In *ICML*, 23–37. Morgan Kaufmann.

Reddy, C., and Tadepalli, P. 1999. Learning Horn definitions: Theory and an application to planning. *New Generation Comput.* 17(1):77–98.

Schlobach, S.; Huang, Z.; Cornet, R.; and Van Harmelen, F. 2007. Debugging incoherent terminologies. *Journal of Automated Reasoning* 39(3):317–349.

Selman, J., and Fern, A. 2011. Learning first-order definite theories via object-based queries. In *ECML/PKDD (3)*, 159–174.

Stuckenschmidt, H.; Parent, C.; and Spaccapietra, S., eds. 2009. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer.

ten Cate, B.; Dalmau, V.; and Kolaitis, P. G. 2012. Learning schema mappings. In *ICDT*, 182–195.

Wang, H.; Horridge, M.; Rector, A.; Drummond, N.; and Seidenberg, J. 2005. Debugging OWL-DL ontologies: A heuristic approach. In *The Semantic Web–ISWC 2005*. Springer. 745–757.