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 polynomial query learnability are open for polynomial time

[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.		✓	✓	✗	✓

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 retrieval* 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).

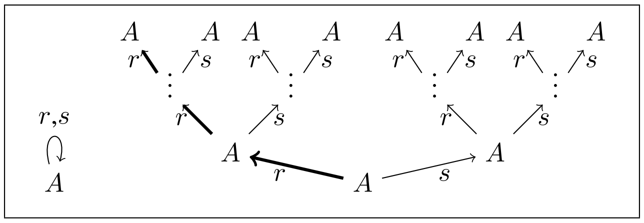


Figure 1: An ABox $\mathcal{A} = \{r(a, a), s(a, a), A(a)\}$ and its unravelling up to level n.

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 $\mathfrak{F}_{\mathcal{D}}(\text{DL-Lite}^{\exists}_{\mathcal{H}}, \mathcal{ELI}\text{-IQ})$ and $\mathfrak{F}_{\mathcal{D}}(\mathcal{ELH}_{\mathsf{lhs}}, \text{AQ})$ by reduction to $\mathfrak{F}_{\mathcal{S}}(\text{DL-Lite}^{\exists}_{\mathcal{H}})$ 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 $\mathfrak{F}_{\mathcal{D}}(\text{DL-Lite}^{\exists}_{\mathcal{H}}, \mathcal{ELI}\text{-IQ})$ 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 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

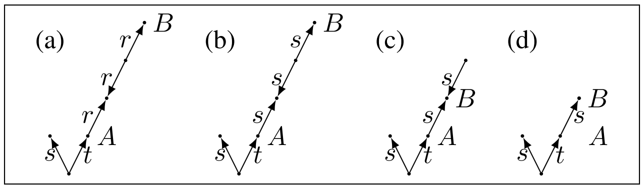


Figure 2: Concept C being role saturated and parent/child merged.

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.*

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

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 $(\mathcal{A}, B(\rho_{\mathcal{A}}))$
```

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

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

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.

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

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.; Ludwig, M.; Walther, D.; and Wolter, F. 2012. The logical difference for the lightweight description logic EL. *J. Artif. Intell. Res. (JAIR)* 44:633–708.

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.

Lutz, C.; Piro, R.; and Wolter, F. 2011. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *IJCAI*, 983–988.

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.

Technical Tools

We start by introducing basic tools for studying the DLs considered in this paper. These include the canonical (also called universal or minimal) model of DL knowledge bases, the well known link between homomorphisms, relational structures and CQ evaluation, and also the translation between tree-shaped interpretations and $\mathcal{ELI}$-IQs. The DLs studied in this paper are fragments of the DL $\mathcal{ELIH}$, where an $\mathcal{ELIH}$ *TBox* consists of a finite set of CIs $C \sqsubseteq D$ with C, D as $\mathcal{ELI}$ concepts and a finite set of RIs $r \sqsubseteq s$, with r, s

roles. The semantics of $\mathcal{ELIH}$ is given by interpretations. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ that assigns each concept name A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. To interpret an ABox $\mathcal{A}$, we consider interpretations $\mathcal{I}$ which also assign to each $a \in \mathsf{Ind}(\mathcal{A})$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, where we assume that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ whenever $a \neq b$ (the *unique name assumption*). The *extension* $C^{\mathcal{I}}$ of an $\mathcal{ELI}$ concept expression C is inductively defined as follows:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e \in C^{\mathcal{I}} \ : \ (d,e) \in r^{\mathcal{I}}\}$
- $(\exists r^{-}.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e \in C^{\mathcal{I}} \ : \ (e,d) \in r^{\mathcal{I}}\}$

An interpretation $\mathcal{I}$ *satisfies*:

- a concept inclusion $C \sqsubseteq D$, in symbols $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$;
- a role inclusion $r \sqsubseteq s$, in symbols $\mathcal{I} \models r \sqsubseteq s$, if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$;
- an instance assertion $C(a)$, in symbols $\mathcal{I} \models C(a)$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$;
- a role assertion $r(a,b)$, in symbols $\mathcal{I} \models r(a,b)$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$.

We say that an interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ (an ABox $\mathcal{A}$) if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T}$ ($\alpha \in \mathcal{A}$). A CI (a RI) α *follows from* a TBox $\mathcal{T}$ if every model of $\mathcal{T}$ is a model of α, in symbols $\mathcal{T} \models \alpha$. We use $\models \alpha$ to denote that α follows from the empty TBox. A knowledge base (KB) is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A query q follows from $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if every model of $(\mathcal{T}, \mathcal{A})$ is a model of q, in symbols $(\mathcal{T}, \mathcal{A}) \models q$. If the DL at hand allows inverse roles then we assume that the ABox $\mathcal{A}$ is closed under inverses, i.e. $r(a,b) \in \mathcal{A}$ iff $r^{-}(b,a) \in \mathcal{A}$. If we add an assertion $r(a,b)$ in $\mathcal{A}$ then we do so assuming that $r^{-}(b,a)$ is also added and, so, the resulting $\mathcal{A}$ is again closed under inverses. Similarly, if an assertion $r(a,b)$ is removed from $\mathcal{A}$ then we do so assuming that $r^{-}(b,a)$ is also removed.

Trees and Homomorphisms A *path* in an $\mathcal{ELI}$ concept expression C is a finite sequence $C_0 \cdot r_1 \cdot C_1 \cdot \ldots \cdot r_k \cdot C_k$, where $C_0 = C$, $k \geq 0$, and $\exists r_{i+1}.C_{i+1}$ is a top-level conjunct of C_i, for $0 \leq i < k$. The set $\mathsf{paths}(C)$ contains all paths in C. We also define $\mathsf{tail}(p) = \{A \mid A$ is a top-level conjunct of $C_k\}$, where C_k is the last concept expression in path p.

Definition 10 *The interpretation $\mathcal{I}_C$ of an $\mathcal{ELI}$ concept expression C is defined as follows:*

- $\Delta^{\mathcal{I}_C} = \mathsf{paths}(C)$
- $A^{\mathcal{I}_C} = \{p \in \mathsf{paths}(C) \mid A \in \mathsf{tail}(p)\}$
- $r^{\mathcal{I}_C} = \{(p, p') \in \mathsf{paths}(C) \times \mathsf{paths}(C) \mid p' = p \cdot r \cdot D\}$

We denote the root of $\mathcal{I}_C$ by ρ_C.

The next lemma relates homomorphisms from interpretations $\mathcal{I}_C$ into interpretations $\mathcal{I}$ to the extension of the concept C in $\mathcal{I}$. Given two interpretations $\mathcal{I}$ and $\mathcal{J}$, a homomorphism $h : \mathcal{I} \to \mathcal{J}$ is a mapping from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{J}}$ such that

- if $d \in A^{\mathcal{I}}$, then $h(d) \in A^{\mathcal{J}}$ for all $A \in \mathsf{N_C}$;
- if $(d, d') \in r^{\mathcal{I}}$, then $(h(d), h(d')) \in r^{\mathcal{J}}$, for all $r \in \mathsf{N_R}$.

The proof of the following lemma is straightforward.

Lemma 11 *Let C be an $\mathcal{ELI}$ concept expression. Let $\mathcal{I}$ be an interpretation with $d \in \Delta^{\mathcal{I}}$. Then, $d \in C^{\mathcal{I}}$ if, and only if, there is a homomorphism $h : \mathcal{I}_C \to \mathcal{I}$ such that $h(\rho_C) = d$.*

Canonical Models for $\mathcal{ELIH}$

We introduce the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ of a knowledge base consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ and the canonical model $\mathcal{I}_{C,\mathcal{T}}$ of an $\mathcal{ELI}$ concept expression C ond TBox $\mathcal{T}$. We start by introducing the canonical model $\mathcal{I}_{\mathcal{A}}$ of an ABox $\mathcal{A}$.

Definition 12 *The canonical model $\mathcal{I}_{\mathcal{A}} = (\Delta^{\mathcal{I}_{\mathcal{A}}}, \cdot^{\mathcal{I}_{\mathcal{A}}})$ of an ABox $\mathcal{A}$ is defined as follows:*

- $\Delta^{\mathcal{I}_{\mathcal{A}}} = \{a \mid a \in \mathsf{Ind}(\mathcal{A})\}$
- $A^{\mathcal{I}_{\mathcal{A}}} = \{a \mid A(a) \in \mathcal{A}, A \in \mathsf{N_C}\}$
- $r^{\mathcal{I}_{\mathcal{A}}} = \{(a,b) \mid r(a,b) \in \mathcal{A}, r \in \mathsf{N_R}\}$

The canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ of an $\mathcal{ELIH}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is defined as the union of a sequence of interpretations $\mathcal{I}_0, \mathcal{I}_1, \ldots$. We define $\mathcal{I}_0$ by extending $\mathcal{I}_{\mathcal{A}}$ with

$$r^{\mathcal{I}_0} = \{(a,b) \mid s(a,b) \in \mathcal{A}, \mathcal{T} \models s \sqsubseteq r\},$$

where r, s are roles. Assume now that $\mathcal{I}_n$ has been defined. Its domain $\Delta^{\mathcal{I}_n}$ consists of sequences $a_0 \cdot r_0 \cdot C_0 \cdot r_1 \cdot C_1 \cdot \ldots \cdot r_m \cdot C_m$, where $a_0 \in \mathsf{Ind}(\mathcal{A})$. To define $\mathcal{I}_{n+1}$, we introduce some notation. For sequences

$$p = a_0 \cdot r_0 \cdot C_0 \cdot r_1 \cdot C_1 \cdot \ldots \cdot r_m \cdot C_m$$

and

$$q = C'_0 \cdot r'_1 \cdot C'_1 \cdot \ldots \cdot r'_{m'} \cdot C'_{m'}$$

we define the concatenation $p \cdot s \cdot q$ of p and q through a role s as

$$a_0 \cdot r_0 \cdot C_0 \cdot r_1 \cdot C_1 \cdot \ldots \cdot r_m \cdot C_m \cdot s \cdot C'_0 \cdot r'_1 \cdot C'_1 \cdot \ldots \cdot r'_{m'} \cdot C'_{m'}$$

Now let $k \leq n$ be minimal such that there is $C \sqsubseteq D \in \mathcal{T}$ and $p \in \Delta^{\mathcal{I}_k}$ with $p \in C^{\mathcal{I}_k}$, but $p \notin D^{\mathcal{I}_n}$. Let D be of the form $\bigsqcap_{1 \leq i \leq l} A_i \sqcap \bigsqcap_{1 \leq j \leq l'} \exists s_j.E_j$, where A_i are concept names; s_j are roles and C_j are $\mathcal{ELI}$ concept expressions with $1 \leq i \leq l$, $1 \leq j \leq l'$; and $l, l' \geq 0$. Then we define $\mathcal{I}_{n+1}$ as follows:

$$\Delta^{\mathcal{I}_{n+1}} = \Delta^{\mathcal{I}_n} \cup \{p \cdot s_j \cdot q \mid q \in \mathsf{paths}(E_j), 1 \leq j \leq l'\};$$

for all $A \in \mathsf{N_C}$:

$$\begin{aligned} A^{\mathcal{I}_{n+1}} \quad = \quad & A^{\mathcal{I}_n} \cup \\ & \{p \cdot s_j \cdot q \mid A \in \mathsf{tail}(q), 1 \leq j \leq l'\} \cup \\ & \{p \mid A_i = A, 1 \leq i \leq l\}; \end{aligned}$$

for all role r:

$$\begin{aligned} r^{\mathcal{I}_{n+1}} \quad = \quad & r^{\mathcal{I}_n} \cup \\ & \{(p \cdot s_j \cdot q, p \cdot s_j \cdot q') \mid (q, q') \in s^{\mathcal{I}_{E_j}}, \\ & \quad \mathcal{T} \models s \sqsubseteq r, 1 \leq j \leq l'\} \cup \\ & \{(p, p \cdot s_j \cdot E_j) \mid \mathcal{T} \models s_j \sqsubseteq r, 1 \leq j \leq l'\}. \end{aligned}$$

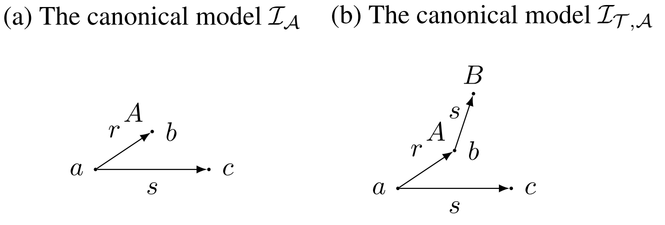

(a) The canonical model $\mathcal{I}_{\mathcal{A}}$
(b) The canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$
A
r
b
a
s
c
B
s
A
r
b
a
s
c

Figure 3: Canonical Models with $\mathcal{A} = \{r(a,b), A(b), s(a,c)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists s.B\}$

This concludes the inductive definition of the sequence $\mathcal{I}_0, \mathcal{I}_1, \ldots$. Finally, we set $\mathcal{I}_{\mathcal{T},\mathcal{A}} = \bigcup_{n\geq 0} \mathcal{I}_n$. As an example let $\mathcal{A} = \{r(a,b), A(b), s(a,c)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists s.B\}$. Figures 3-a and 3-b show the interpretations $\mathcal{I}_{\mathcal{A}}$ and $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, respectively.

The following two lemmas summarize the main properties of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. The proofs are straightforward.

Lemma 13 *Let q be a CQ and $(\mathcal{T}, \mathcal{A})$ an $\mathcal{ELIH}$ knowledge base. Then $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q$ if and only if $(\mathcal{T}, \mathcal{A}) \models q$.*

The canonical model $\mathcal{I}_{C,\mathcal{T}}$ of an $\mathcal{ELI}$ concept expression C and a $\mathcal{ELIH}$ TBox $\mathcal{T}$ is defined as $\mathcal{I}_{\mathcal{T},\mathcal{A}_C}$.

Lemma 14 *Let C and D be $\mathcal{ELI}$ concept expressions and $\mathcal{T}$ an $\mathcal{ELIH}$ TBox. Then $\mathcal{I}_{C,\mathcal{T}} \models D(\rho_C)$ if, and only if, $\mathcal{T} \models C \sqsubseteq D$.*

Definition of Polynomial Time Learnability

We employ the following standard definition of polynomial time exact learnability. A learning framework $(X, \mathcal{L}, \mu)$ is polynomial *time* exact 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 is the target and $x \in X$ is the largest counterexample seen so far.

The following proposition is a direct consequence of fact that any polynomial time algorithm only generates polynomial size output.

Proposition 15 *If a learning framework $\mathfrak{F}$ is polynomially time exact learnable then $\mathfrak{F}$ is polynomially query exact learnable.*

In what follows, whenever possible, we prove polynomial time learnability for the positive results in Table 1. By Proposition 15 this implies polynomial query learnability.

Proofs for Theorem 1 and Theorem 9

It remains to prove polynomial time exact learnability of $\mathcal{ELH}_{\mathsf{lhs}}$ TBoxes in the subsumption framework. To this end, we extend the polynomial time learnability proof for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes presented in (Konev et al. 2014) to $\mathcal{ELH}_{\mathsf{lhs}}$ TBoxes. The main challenge in allowing role inclusions is that the product construction, which is fundamental for the learning algorithm presented in (Konev et al. 2014), has to take into account the role hierarchy. In particular, the product construction can lead to non-tree shaped interpretations which may not be easily mapped into polynomial size tree interpretations as was done in the construction for $\mathcal{EL}_{\mathsf{lhs}}$.

We start by giving a brief overview of the algorithm provided in (Konev et al. 2014), show that two naive attempts to extend it with role inclusions fail and then demonstrate how it can be modified. Before that, we need to introduce some notions.

We often work with interpretations which have the structure of a tree. A directed graph G is a pair (V, E) where V is a set of vertices and $E \subseteq V \times V$ is a set of ordered pairs of vertices (called edges) connecting vertices. A path in a directed graph $G = (V, E)$ is a finite sequence $d_0 \cdot d_1 \cdot \ldots \cdot d_k$, $k \geq 0$, where $(d_i, d_{i+1}) \in E$, for $0 \leq i < k$. The set $\mathsf{paths}(G, d)$ contains all paths in G starting from $d \in V$. That is, if $d_0 \cdot d_1 \cdot \ldots \cdot d_k \in \mathsf{paths}(G, d)$ then $d_0 = d$. Let $\mathsf{tail}_G(p) = d_k$ be the last element in path $p = d_0 \cdot d_1 \cdot \ldots \cdot d_k$. A directed graph G is *tree shaped* if there is a unique element (the root), denoted by ρ_G, such that (i) for every $d \in V$ there is $p \in \mathsf{paths}(G, \rho_G)$ such that $d = \mathsf{tail}_G(p)$ and (ii) for every distinct $p_1, p_2 \in \mathsf{paths}(G, \rho_G)$, we have that $\mathsf{tail}_G(p_1) \neq \mathsf{tail}_G(p_2)$.

Definition 16 (Tree shaped interpretation) *An interpretation $\mathcal{I}$ is* tree shaped *if the directed graph $G_{\mathcal{I}} = (\Delta^{\mathcal{I}}, \{(d, d') \mid \exists r \in \mathsf{N_R}, (d, d') \in r^{\mathcal{I}}\})$ is tree shaped and $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$ for distinct $r, s \in \mathsf{N_R}$.*

Given a tree shaped interpretation $\mathcal{I}$, we denote by $\rho_{\mathcal{I}}$ the unique element of $\Delta^{\mathcal{I}}$ that is the root of $G_{\mathcal{I}}$. Every tree shaped interpretation $\mathcal{I}$ can be viewed as an $\mathcal{EL}$ concept expression $C_{\mathcal{I}}$ in a straightforward way. Also, the tree interpretation $\mathcal{I}_C$ of an $\mathcal{EL}$ concept expression C (Definition 10) is tree shaped.

Definition 17 (Product) *The* product *of two interpretations $\mathcal{I}$ and $\mathcal{J}$ is the interpretation $\mathcal{I} \times \mathcal{J}$ with*

- $\Delta^{\mathcal{I}\times\mathcal{J}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$;
- $(d, e) \in A^{\mathcal{I}\times\mathcal{J}}$ *if* $d \in A^{\mathcal{I}}$ *and* $e \in A^{\mathcal{J}}$;
- $((d, e), (d', e')) \in r^{\mathcal{I}\times\mathcal{J}}$ *if* $(d, d') \in r^{\mathcal{I}}$ *and* $(e, e') \in r^{\mathcal{J}}$.

One can show that the product of tree shaped interpretations is a disjoint union of tree shaped interpretations. If $\mathcal{I}$ and $\mathcal{J}$ are tree shaped interpretations, we denote by $\mathcal{I} \times_\rho \mathcal{J}$ the maximal tree shaped interpretation that is contained in $\mathcal{I} \times \mathcal{J}$ with root $(\rho_{\mathcal{I}}, \rho_{\mathcal{J}})$. Products preserve the truth of $\mathcal{EL}$ concept expressions (Lutz, Piro, and Wolter 2011):

Lemma 18 *For all $\mathcal{EL}$ concepts C: $d \in C^{\mathcal{I}}$ and $e \in C^{\mathcal{J}}$ iff $(d, e) \in C^{\mathcal{I}\times\mathcal{J}}$.*

Definition 19 (Simulation) *Let $\mathcal{I}, \mathcal{J}$ be interpretations, $d_0 \in \Delta^{\mathcal{I}}$ and $e_0 \in \Delta^{\mathcal{J}}$. A relation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ is a* simulation *from $(\mathcal{I}, d_0)$ to $(\mathcal{J}, e_0)$ if $(d_0, e_0) \in S$ and the following conditions are satisfied:*

- *for all concept names $A \in \mathsf{N_C}$ and all $(d, e) \in S$, if $d \in A^{\mathcal{I}}$ then $e \in A^{\mathcal{J}}$;*
- *for all role names $r \in \mathsf{N_R}$, all $(d, e) \in S$ and all $d' \in \Delta^{\mathcal{I}}$, if $(d, d') \in r^{\mathcal{I}}$ then there exists $e' \in \Delta^{\mathcal{J}}$ such that $(e, e') \in r^{\mathcal{J}}$ and $(d', e') \in S$.*

Algorithm 4 The learning algorithm for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes.

```
1:  Let $\mathfrak{I} = \emptyset$ and $\mathcal{H} = \emptyset$
2:  while $\mathcal{H} \not\equiv \mathcal{T}$ do
3:      Let $C \sqsubseteq A$ be the returned positive counterexample
                for $\mathcal{T}$ relative to $\mathcal{H}$
4:      Find a $\mathcal{T}$-essential $\mathcal{T}$-countermodel $\mathcal{I}$ for $\mathcal{H}$
5:      if there is a $\mathcal{J} \in \mathfrak{I}$ such that $\mathcal{J} \not\Rightarrow (\mathcal{I} \times_\rho \mathcal{J})$ and
                $\mathcal{I} \times_\rho \mathcal{J} \not\models \mathcal{T}$ then
6:          Let $\mathcal{J}$ be the first such element of $\mathfrak{I}$
7:          Find $\mathcal{T}$-essential $\mathcal{T}$-countermodel $\mathcal{J}' \subseteq \mathcal{I} \times_\rho \mathcal{J}$ for $\mathcal{H}$
8:          Replace $\mathcal{J}$ in $\mathfrak{I}$ with $\mathcal{J}'$
9:      else
10:         Append $\mathcal{I}$ to $\mathfrak{I}$
11:     Set $\mathcal{H} = \{C_\mathcal{I} \sqsubseteq A \mid \mathcal{I} \in \mathfrak{I}, \mathcal{T} \models C_\mathcal{I} \sqsubseteq A\}$
```

We write $(\mathcal{I}, d_0) \Rightarrow (\mathcal{J}, e_0)$ if there is a simulation from $(\mathcal{I}, d_0)$ to $(\mathcal{J}, e_0)$. If $\mathcal{I}$ and $\mathcal{J}$ are tree shaped then we write $\mathcal{I} \Rightarrow \mathcal{J}$ as a shorthand for $(\mathcal{I}, \rho_\mathcal{I}) \Rightarrow (\mathcal{J}, \rho_\mathcal{J})$, where $\rho_\mathcal{I}$ and $\rho_\mathcal{J}$ are the roots of $\mathcal{I}$ and $\mathcal{J}$, respectively. Simulations preserve the membership to $\mathcal{EL}$ concept expressions (Lutz, Piro, and Wolter 2011):

Lemma 20 *For all $\mathcal{EL}$ concept expressions C: if $d \in C^\mathcal{I}$ and $(\mathcal{I}, d) \Rightarrow (\mathcal{J}, e)$, then $e \in C^\mathcal{J}$.*

We can now give an overview of Algorithm 4 presented in (Konev et al. 2014) to learn $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes . Algorithm 4 maintains a sequence $\mathfrak{I}$ of tree shaped interpretations that intuitively represents the TBox $\mathcal{H}$ constructed in Line 11. In each iteration of the main loop (Line 2), if the hypothesis $\mathcal{H}$ is not equivalent to the target $\mathcal{T}$ then the oracle returns an $\mathcal{EL}_{\mathsf{lhs}}$ CI $C \sqsubseteq A$ that is a positive counterexample. The assumption that $C \sqsubseteq A$ is a positive counterexample is justified by the construction of $\mathcal{H}$, which ensures that at all times $\mathcal{T} \models \mathcal{H}$. A $\mathcal{T}$-countermodel is an interpretation that satisfies $\mathcal{T}$ but not $\mathcal{H}$ or vice-versa. In Line 4, Algorithm 4 takes the tree interpretation $\mathcal{I}_C$ of C and computes a $\mathcal{T}$-countermodel $\mathcal{I}$ with some properties which are defined in (Konev et al. 2014) as '$\mathcal{T}$-essential' (we do not enter into details here). This $\mathcal{T}$-essential $\mathcal{T}$-countermodel $\mathcal{I}$ is either used to refine an element of $\mathfrak{I}$ (Line 8), or it is appended to $\mathfrak{I}$ (Line 10). The conditions for refinement are given in Line 5. One wants to refine an element $\mathcal{J}$ in $\mathfrak{I}$ if the product of $\mathcal{I}$ and $\mathcal{J}$ 'maximizes' $\mathcal{J}$ and also does not satisfy a CI in $\mathcal{T}$, where the condition $\mathcal{I} \times_\rho \mathcal{J} \not\models \mathcal{T}$ is implemented by posing queries of the form $\mathcal{T} \models C_\times \sqsubseteq A$ with $C_\times$ as the concept expression corresponding to $\mathcal{I} \times_\rho \mathcal{J}$. After updating $\mathfrak{I}$, the algorithm returns to the main loop and asks again whether $\mathcal{H} \equiv \mathcal{T}$.

We know that an $\mathcal{ELH}_{\mathsf{lhs}}$ TBox consists of a finite set of RIs and a finite set of $\mathcal{EL}_{\mathsf{lhs}}$ CIs. Since the learner knows the signature $\Sigma_\mathcal{T}$ of the target $\mathcal{T}$, it can learn all role inclusions in $\mathcal{T}$ by simply posing membership queries of the form $\mathcal{T} \models r \sqsubseteq s$, for all $r, s \in \Sigma_\mathcal{T}$.

Suppose that we know the RIs in $\mathcal{T}$ and try to use Algorithm 4 without any modifications to learn the $\mathcal{EL}_{\mathsf{lhs}}$ CIs in $\mathcal{T}$. A simple example shows why this algorithm may *not* be polynomial in the presence of role inclusions. Let $\mathcal{T} = \{\exists s_1^n.\top \sqcap \exists s_2^n.\top \sqsubseteq A, r_1 \sqsubseteq s_1, r_1 \sqsubseteq s_2, r_2 \sqsubseteq s_1, r_2 \sqsubseteq s_2\}$, where s_i^n is an abbreviation for nesting s_i existentials n times, $i \in \{1, 2\}$. Suppose that the hypothesis initially contains

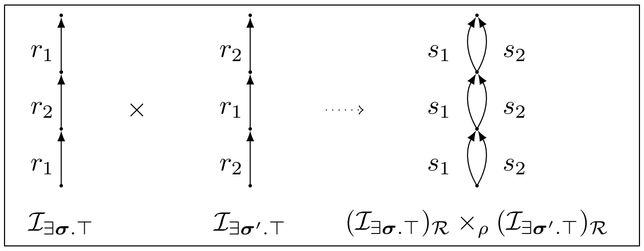

r_1
r_2
r_1
$\times$
r_2
r_1
r_2
s_1
s_2
s_1
s_2
s_1
s_2
$\mathcal{I}_{\exists\boldsymbol{\sigma}.\top}$
$\mathcal{I}_{\exists\boldsymbol{\sigma}'.\top}$
$(\mathcal{I}_{\exists\boldsymbol{\sigma}.\top})_\mathcal{R} \times_\rho (\mathcal{I}_{\exists\boldsymbol{\sigma}'.\top})_\mathcal{R}$

Figure 4: Product of $(\mathcal{I}_{\exists\boldsymbol{\sigma}.\top})_\mathcal{R} \times_\rho (\mathcal{I}_{\exists\boldsymbol{\sigma}'.\top})_\mathcal{R}$ with $\boldsymbol{\sigma} = r_1 r_2 r_1$, $\boldsymbol{\sigma}' = r_2 r_1 r_2$ and $\mathcal{R} = \{r_1 \sqsubseteq s_1, r_1 \sqsubseteq s_2, r_2 \sqsubseteq s_1, r_2 \sqsubseteq s_2\}$.

$\{r_1 \sqsubseteq s_1, r_1 \sqsubseteq s_2, r_2 \sqsubseteq s_1, r_2 \sqsubseteq s_2\}$. For any sequence $\boldsymbol{\sigma} = \sigma^1\sigma^2 \dots \sigma^n$ with $\sigma^j \in \{r_1, r_2\}$, $1 \leq j \leq n$, the concept expression $\exists\boldsymbol{\sigma}.\top$ stands for $\exists\sigma^1.\exists\sigma^2.\dots.\exists\sigma^n.\top$. Then the oracle can give as counterexample $\exists\boldsymbol{\sigma}.\top \sqsubseteq A$, where $\boldsymbol{\sigma}$ is any of the 2^n sequences of length n. Since the product of $\mathcal{I}_{\exists\boldsymbol{\sigma}.\top}$ and $\mathcal{I}_{\exists\boldsymbol{\sigma}'.\top}$ for any $\boldsymbol{\sigma} \neq \boldsymbol{\sigma}'$ would correspond to a concept expression $C_\times$ such that $\mathcal{T} \not\models C_\times \sqsubseteq A$, Algorithm 4 would append all these 2^n interpretations to $\mathfrak{I}$.

The reason why Algorithm 4 does not work with role inclusions is that, by not taking into account the role hierarchy, we do not refine elements of $\mathfrak{I}$ not satisfying the same CI in $\mathcal{T}$ (in this example $\exists s_1^n.\top \sqcap \exists s_2^n.\top \sqsubseteq A$). To have a polynomial bound on the size of $\mathfrak{I}$, we want to ensure that at all times each CI in $\mathcal{T}$ is not satisfied by at most one element $\mathcal{J}$ in $\mathfrak{I}$.

We now modify the algorithm above to learn $\mathcal{T}$ by also taking into account the role hierarchy and by refining the notion of a $\mathcal{T}$-essential interpretation. For this purpose, we introduce the notion of *role completion* $\mathcal{I}_\mathcal{R}$ of an interpretation $\mathcal{I}$ w.r.t. a set $\mathcal{R}$ of RIs in $\mathcal{T}$.

Definition 21 (Role Completion) *Let $\mathcal{R}$ be a set of RIs and $\mathcal{I}$ be an interpretation. The role completion $\mathcal{I}_\mathcal{R} = (\Delta^{\mathcal{I}_\mathcal{R}}, \cdot^{\mathcal{I}_\mathcal{R}})$ of $\mathcal{I}$ w.r.t. $\mathcal{R}$ is defined as follows:*

- $\Delta^{\mathcal{I}_\mathcal{R}} := \Delta^\mathcal{I}$
- $d \in A^{\mathcal{I}_\mathcal{R}}$ *iff* $d \in A^\mathcal{I}$
- $s^{\mathcal{I}_\mathcal{R}} := \{(d, e) \mid (d, e) \in r^\mathcal{I}, \mathcal{R} \models r \sqsubseteq s\}$

Observe that $\mathcal{I}_\mathcal{R}$ is polynomially computable in $|\mathcal{I}|$ and $|\mathcal{R}|$. If $\mathcal{I}$ and $\mathcal{J}$ do not satisfy some CI in $\mathcal{T}$ then the product of their role completions also does not satisfy it. In fact, we can see that in our example, for any $\boldsymbol{\sigma} \neq \boldsymbol{\sigma}'$, the product of $(\mathcal{I}_{\exists\boldsymbol{\sigma}.\top})_\mathcal{R}$ and $(\mathcal{I}_{\exists\boldsymbol{\sigma}'.\top})_\mathcal{R}$ does not satisfy $\exists s_1^n.\top \sqcap \exists s_2^n.\top \sqsubseteq A$. Though, given two tree shaped interpretations $\mathcal{I}$ and $\mathcal{J}$, the product of their role completions may not be tree shaped (see Figure 4). Because of the RIs, the resulting structure is what can be called *multi-edge* tree shaped.

A multi-edge tree shaped interpretation is a generalization of a tree shaped interpretation where one allows elements to be connected by multiple roles. Figure 5 illustrates interpretations which we classify as:

- tree shaped (left); and
- multi-edge tree shaped (right).

Definition 22 *An interpretation $\mathcal{I}$ is* multi-edge tree shaped *if the directed graph $G_\mathcal{I} = (\Delta^\mathcal{I}, \{(d, e) \mid \exists r \in \mathsf{N_R}, (d, e) \in r^\mathcal{I}\})$ is tree shaped.*

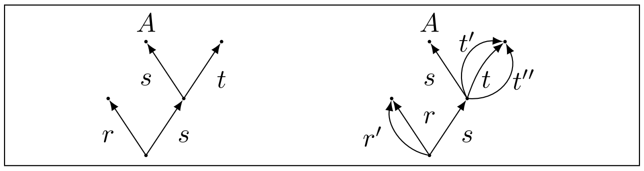


Figure 5: Tree shaped (left) and multi-edge tree shaped interpretation (right).

In what follows we often call tree shaped interpretations *single-edge tree shaped* to emphasize the difference to multi-edge tree shaped interpretations. Note that if $\mathcal{I}$ is multi-edge tree shaped then $\mathcal{I}_\mathcal{R}$ is also multi-edge tree shaped.

For our second attempt to learn $\mathcal{ELH}_{\mathsf{lhs}}$ TBoxes, we need to introduce a few more notions. Let $\mathcal{I}$ be a multi-edge tree shaped interpretation. We denote by

- $\mathcal{I}^{-r(d,e)}$ the result of removing the pair (d, e) from $r^\mathcal{I}$;
- $\mathcal{I}^{-d\downarrow}$ the result of removing the subtree rooted at d from $\mathcal{I}$;
- $\mathcal{I}^{-\rho_\mathcal{I}}$ the result of removing the root $\rho_\mathcal{I}$ from $\mathcal{I}$.

Definition 23 (Multi-edge reduced) *An interpretation $\mathcal{I}$ is called* multi-edge reduced *if the preconditions of the following two rules do not apply to $\mathcal{I}$:*

- *(Remove Subtrees) If $\mathcal{I}_\mathcal{R} \Rightarrow (\mathcal{I}^{-d\downarrow})_\mathcal{R}$, then replace $\mathcal{I}$ by $\mathcal{I}^{-d\downarrow}$.*
- *(Remove Pairs of Elements) If $\mathcal{I}_\mathcal{R} = (\mathcal{I}^{-r(d,e)})_\mathcal{R}$, then replace $\mathcal{I}$ by $\mathcal{I}^{-r(d,e)}$.*

Intuitively, in the definition above, the first rule removes 'redundant' elements from $\Delta^\mathcal{I}$ and the second rule removes 'redundant' role assertions from $\mathcal{I}$. If $\mathcal{I}$ and $\mathcal{J}$ are multi-edge tree shaped interpretations, we denote by $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R}$ the *maximal multi-edge reduced tree shaped interpretation* that is contained in $\mathcal{I}_\mathcal{R} \times \mathcal{J}_\mathcal{R}$ with root $(\rho_\mathcal{I}, \rho_\mathcal{J})$. The following lemma is immediate from the definition of the rules above.

Lemma 24 *Let $\mathcal{J}$ be the result of applying the rules (Remove subtrees) and (Remove Pairs of Elements) exhaustively to $\mathcal{I}$. Then, $\mathcal{J}_\mathcal{R} \Rightarrow \mathcal{I}_\mathcal{R}$ and $\mathcal{I}_\mathcal{R} \Rightarrow \mathcal{J}_\mathcal{R}$.*

An interpretation $\mathcal{I}$ is an $(\mathcal{T}, \mathcal{R})$*-countermodel* for $\mathcal{H}$ if $\mathcal{I}_\mathcal{R} \models \mathcal{H}$ and $\mathcal{I}_\mathcal{R} \not\models \mathcal{T}$. We describe a class of $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ that can be viewed as a variant of the $\mathcal{T}$-essential $\mathcal{T}$-countermodels for $\mathcal{H}$ studied in (Konev et al. 2014).

Definition 25 *A $(\mathcal{T}, \mathcal{R})$-countermodel $\mathcal{I}$ for $\mathcal{H}$ is $\mathcal{T}$-essential if the following conditions are satisfied:*

1. *$\mathcal{I}$ is single-edge tree shaped;*
2. *$(\mathcal{I}^{-\rho_\mathcal{I}})_\mathcal{R} \models \mathcal{T}$;*
3. *$(\mathcal{I}^{-d\downarrow})_\mathcal{R} \models \mathcal{T}$ for all $d \in \Delta^\mathcal{I} \setminus \{\rho_\mathcal{I}\}$.*

It follows from Points 1 and 3 of Definition 25 that $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ do not have redundant role assertions nor redundant subtrees, which implies that such interpretations are already multi-edge reduced.

Lemma 26 *$\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ are multi-edge reduced.*

Now, consider Algorithm 5, which is very similar to Algorithm 4 for $\mathcal{EL}_{\mathsf{lhs}}$ (this time taking into account the role hierarchy). We assume that at each state in the run of the learning algorithm, the hypothesis $\mathcal{H}$ of the learner is of the form $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$, where $\mathcal{H}_{basic}$ contains the set $\mathcal{R}$ of RIs in $\mathcal{T}$ and $\mathcal{H}_{add}$ contains only $\mathcal{EL}_{\mathsf{lhs}}$ CIs. As illustrated by Figure 4, even if multi-edge reduced, the product of the role completion of tree shaped interpretations may not be single-edge tree shaped. Since membership queries receive as input subsumptions $C \sqsubseteq A$ and the underlying structure of an $\mathcal{EL}$ concept expression C is single-edge tree shaped, we need our interpretations to be of this form. Though, simply unfolding a multi-edge tree can result in an exponentially larger single-edge tree. So, in Line 7, we additionally check whether the product $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R}$ is single-edge tree shaped.

This algorithm may *also not* be polynomial in the presence of RIs. To illustrate, consider again that $\mathcal{T} = \{\exists s_1^n.\top \sqcap \exists s_2^n.\top \sqsubseteq A, r_1 \sqsubseteq s_1, r_1 \sqsubseteq s_2, r_2 \sqsubseteq s_1, r_2 \sqsubseteq s_2\}$. Suppose that the hypothesis initially contains $\{r_1 \sqsubseteq s_1, r_1 \sqsubseteq s_2, r_2 \sqsubseteq s_1, r_2 \sqsubseteq s_2\}$. Then the oracle can give as counterexample $\exists\boldsymbol{\sigma}.\top \sqsubseteq A$, where $\boldsymbol{\sigma}$ is any of the 2^n sequences of r_1s and r_2s of length n. Note that the product of the role completion of the tree interpretations of any two concepts $\exists\boldsymbol{\sigma}.\top$ and $\exists\boldsymbol{\sigma}'.\top$ would not be single-edge tree shaped. Then, Algorithm 5 would append all of these counterexamples.

Algorithm 5 Naive learning algorithm for $\mathcal{ELH}_{\mathsf{lhs}}$ TBoxes

```
1:  Let $\mathfrak{I} = \emptyset$ and $\mathcal{H}_{add} = \emptyset$
2:  Set $\mathcal{H}_{basic} = \{r \sqsubseteq s \mid \mathcal{T} \models r \sqsubseteq s$ where $r, s \in \mathsf{N_R}\}$
3:  Set $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$
4:  while $\mathcal{H} \not\equiv \mathcal{T}$ do
5:      Let $C \sqsubseteq A$ be the returned positive counterexample
                                              for $\mathcal{T}$ relative to $\mathcal{H}$
6:      Find a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel $\mathcal{I}$ for $\mathcal{H}$
7:      if there is a $\mathcal{J} \in \mathfrak{I}$ such that $\mathcal{J}_\mathcal{R} \not\Leftarrow (\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R})$,
                  $(\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R})$ is single-edge tree shaped and
                                      $(\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R}) \not\models \mathcal{T}$ then
8:          Let $\mathcal{J}$ be the first such element of $\mathfrak{I}$
9:          Find a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel
                              $\mathcal{J}' \subseteq (\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R})$ for $\mathcal{H}$
10:         Replace $\mathcal{J}$ in $\mathfrak{I}$ with $\mathcal{J}'$
11:     else
12:         Append $\mathcal{I}$ to $\mathfrak{I}$
13:     Set $\mathcal{H}_{add} = \{C_\mathcal{J} \sqsubseteq A \mid \mathcal{J} \in \mathfrak{I}, \mathcal{T} \models C_\mathcal{J} \sqsubseteq A\}$
14:     Set $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$
```

The reason why Algorithm 5 also does not work is that, by not replacing $\mathcal{J}$ when $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R}$ is not single-edge tree shaped, we may have exponentially many elements $\mathcal{J} \in \mathfrak{I}$ such that $\mathcal{J}_\mathcal{R}$ does not satisfy the same CI in $\mathcal{T}$ (in this example $\exists s_1^n.\top \sqcap \exists s_2^n.\top \sqsubseteq A$).

We finally solve this in Algorithm 6 by calling the recursive function 'RefineCounterModel' (Line 7), which refines the $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel $\mathcal{I}$ with the elements of $\mathfrak{I}$ before updating $\mathfrak{I}$ with $\mathcal{I}$. Whenever $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R}$ is not single-edge tree shaped (otherwise we are done), we compute an interpretation that we call '$\mathcal{I}$-candidate for $\mathcal{J}$' (Line 11 of Algorithm 7). An $\mathcal{I}$-candidate $\mathcal{I}^\times$ for $\mathcal{J}$ can be thought of as an 'intermediate' between $\mathcal{I}$ and $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R}$ which can be transformed in polynomial time into a single-edge tree

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

```
 1: Let $\mathfrak{I} = \emptyset$ and $\mathcal{H}_{add} = \emptyset$
 2: Set $\mathcal{H}_{basic} = \{r \sqsubseteq s \mid \mathcal{T} \models r \sqsubseteq s$ where $r, s \in \mathsf{N_R}\}$
 3: Set $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$
 4: while $\mathcal{H} \not\equiv \mathcal{T}$ do
 5:     Let $i = 0$
 6:     Let $C \sqsubseteq A$ be the returned positive counterexample
                                        for $\mathcal{T}$ relative to $\mathcal{H}$
 7:     $(\mathcal{I}, i)$ :=RefineCounterModel($\mathcal{I}_C$, $i$)
 8:     if $i \geq 1$ then
 9:         Let $\mathcal{J}^i$ be the element at position $i$ in $\mathfrak{I}$
10:         Find a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel
                         $\mathcal{J}' \subseteq (\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^i_\mathcal{R})$ for $\mathcal{H}$
11:         Replace $\mathcal{J}^i$ in $\mathfrak{I}$ with $\mathcal{J}'$
12:     else
13:         Append $\mathcal{I}$ to $\mathfrak{I}$
14:     Set $\mathcal{H}_{add} = \{C_\mathcal{J} \sqsubseteq A \mid \mathcal{J} \in \mathfrak{I}, \mathcal{T} \models C_\mathcal{J} \sqsubseteq A\}$
15:     Set $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$
```

Algorithm 7 Refining a $(\mathcal{T}, \mathcal{R})$-countermodel $\mathcal{I}$ for $\mathcal{H}$

```
 1: function RefineCounterModel($\mathcal{K}$, $i$)
 2:     Find $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel $\mathcal{I} \subseteq \mathcal{K}^{(\mathcal{R})}_\mathcal{H}$ for $\mathcal{H}$
 3:     Let $j = 1$
 4:     while $j \leq |\mathfrak{I}|$ do
 5:         Let $\mathcal{J}^j$ be the element at position $j$ in $\mathfrak{I}$
 6:         if $\mathcal{J}^j_\mathcal{R} \not\to (\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R})$ then
 7:             if $(\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R}) \not\models \mathcal{T}$ and $(\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R})$
                                  is single-edge tree shaped then
 8:                 $i = j$
 9:                 return $(\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R}, i)$
10:             else
11:                 Find a single-edge tree shaped
                                  $\mathcal{I}$-candidate $\mathcal{I}^\times$ for $\mathcal{J}^j$
12:                 if $\mathcal{I}^\times$ is such that $\mathcal{I}^\times_\mathcal{R} \not\models \mathcal{T}$ then
13:                     $(\mathcal{I}, i)$ :=RefineCounterModel($\mathcal{I}^\times$, $i$)
14:                     return $(\mathcal{I}, i)$
15:         $j = j + 1$
16:     return $(\mathcal{I}, i)$
```

shaped interpretation.

We can detect whether there is $\mathcal{J}$ which needs to be replaced by posing queries of the form $C_{\mathcal{I}^\times} \sqsubseteq A$. However, we cannot use $\mathcal{I}^\times$ to replace $\mathcal{J} \in \mathfrak{I}$. The reason is that we may not find a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel from a single-edge tree shaped $\mathcal{I}$-candidate for $\mathcal{J}$ just by removing subtrees (a property we show we would have for single-edge products $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R}$). So we recursively call 'RefineCounterModel' with this $\mathcal{I}$-candidate $\mathcal{I}^\times$ for $\mathcal{J}$ as input.

To bound the number of recursive calls, $\mathcal{I}^\times$ needs to be a $(\mathcal{T}, \mathcal{R})$-countermodel that is strictly 'more general' than the previous $\mathcal{I}$. Since we can show that the number of recursive calls is bounded, Algorithm 7 enters the 'if' in Line 7 if the conditions for replacement (1) $\mathcal{J}_\mathcal{R} \not\to (\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R})$ and (2) $(\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R}) \not\models \mathcal{T}$ are satisfied, returns from the function 'RefineCounterModel' and replaces $\mathcal{J}$, as required (a variable i indicates the position in $\mathfrak{I}$ where an element $\mathcal{J}$ should be replaced in Algorithm 6).

The notion of an $\mathcal{I}$-candidate for $\mathcal{J}$ is defined as follows. For multi-edge tree shaped interpretations $\mathcal{I}$ and $\mathcal{J}$, if there is a homomorphism $h : \mathcal{I} \to \mathcal{J}$ (mapping the respective roots) and $\mathcal{J} \not\to \mathcal{I}$ then we write $\mathcal{I} \overset{\not\leftarrow}{\to} \mathcal{J}$.

Definition 27 *An interpretation $\mathcal{I}^\times$ is an $\mathcal{I}$-candidate for $\mathcal{J}$ if the following conditions are satisfied:*

1. $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R} \overset{\not\leftarrow}{\to} \mathcal{I}^\times_\mathcal{R} \overset{\not\leftarrow}{\to} \mathcal{I}_\mathcal{R}$*;*
2. $\mathcal{I}^\times_\mathcal{R} \models \mathcal{H}$*.*

Intuitively, Point (2) says that $\mathcal{I}^\times$ is strictly 'more general' than $\mathcal{I}$ and strictly 'more restricted' than $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R}$.

Recall the scenario described in the example above. Let $\mathcal{T} = \{\exists s_1^n.\top \sqcap \exists s_2^n.\top \sqsubseteq A, r_1 \sqsubseteq s_1, r_1 \sqsubseteq s_2, r_2 \sqsubseteq s_1, r_2 \sqsubseteq s_2\}$ and $\mathcal{H} = \{r_1 \sqsubseteq s_1, r_1 \sqsubseteq s_2, r_2 \sqsubseteq s_1, r_2 \sqsubseteq s_2\}$. Suppose the oracle chooses a sequence $\boldsymbol{\sigma}$ of length n and returns the counterexample $\exists \boldsymbol{\sigma}.\top \sqsubseteq A$, with $\boldsymbol{\sigma} = \sigma^1\sigma^2 \ldots \sigma^n$ and $\sigma^j \in \{r_1, r_2\}$, $1 \leq j \leq n$. Algorithm 6 calls 'RefineCounterModel' with $(\mathcal{I}_{\exists\boldsymbol{\sigma}.\top}, 0)$ as input. As $\mathfrak{I}$ is initially empty, we return from 'RefineCounterModel' with $(\mathcal{I}_{\exists\boldsymbol{\sigma}.\top}, 0)$. Note that $\mathcal{I}_{\exists\boldsymbol{\sigma}.\top}$ is already a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel. Then, Algorithm 6 appends $\mathcal{I}_{\exists\boldsymbol{\sigma}.\top}$ to $\mathfrak{I}$. Suppose that in the second iteration the oracle chooses a sequence $\boldsymbol{\sigma}' \neq \boldsymbol{\sigma}$ and returns the counterexample $\exists\boldsymbol{\sigma}'.\top \sqsubseteq A$. Now, Algorithm 6 calls 'RefineCounterModel' with $(\mathcal{I}_{\exists\boldsymbol{\sigma}'.\top}, 0)$ as input. Since $(\mathcal{I}_{\exists\sigma.\top})_\mathcal{R} \not\to ((\mathcal{I}_{\exists\sigma.\top})_\mathcal{R} \times_\rho (\mathcal{I}_{\exists\sigma'.\top})_\mathcal{R})$ and $((\mathcal{I}_{\exists\sigma.\top})_\mathcal{R} \times^{\mathsf{r}}_\rho (\mathcal{I}_{\exists\sigma'.\top})_\mathcal{R})$ is not single-edge tree shaped we compute a single-edge tree shaped $\mathcal{I}_{\exists\boldsymbol{\sigma}'.\top}$-candidate for $\mathcal{I}_{\exists\boldsymbol{\sigma}.\top}$, which could be $\mathcal{I}^\times_C$ with $C = \exists\sigma^1.\exists\sigma^2.\ldots\exists\sigma^{n-1}.(\exists s_1.\top \sqcap \exists s_2.\top)$, if $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ differ in the last role. Then, we make a recursive call to 'RefineCounterModel' with $(\mathcal{I}^\times_C, 0)$ as input. To simplify the example suppose that $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ only differ in the last role. Then, $\mathcal{I}^\times_C = (\mathcal{I}^\times_C)_\mathcal{R} \times^{\mathsf{r}}_\rho (\mathcal{I}_{\exists\boldsymbol{\sigma}.\top})_\mathcal{R}$. As $\mathcal{I}^\times_C$ is single-edge tree shaped we would enter in the 'if' in Line 7 of Algorithm 7, return from 'RefineCounterModel' with $(\mathcal{I}^\times_C, 1)$ and replace $\mathcal{I}_{\exists\boldsymbol{\sigma}.\top}$ in $\mathfrak{I}$ (Line 11 of Algorithm 6).

The intuition for why Algorithm 6 works is that, by computing 'intermediate' interpretations, we can detect whether there is an element of $\mathfrak{I}$ which should be replaced. By making the replacements correctly, we can ensure that each CI in $\mathcal{T}$ is not satisfied by at most one $\mathcal{J} \in \mathfrak{I}$.

We now formally show that Algorithm 6 runs in polynomial time and learns $\mathcal{ELH}_{\mathsf{lhs}}$ TBoxes. Clearly, if Algorithm 6 terminates, then the output is equivalent to the target TBox. Thus, it is sufficient to show that the learning algorithm terminates in polynomial time with respect to the size of the target TBox $\mathcal{T}$ and the largest counterexample received so far.

$\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels

First we present some properties of $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$. These properties are:

- $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ are polynomially computable (Lemma 29);
- $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ have size bounded by the target TBox (Lemma 30);
- if $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R}$ is single-edge tree shaped then it can be transformed into a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$ by removing subtrees (Lemma 31).

Before we show the three properties listed above, we introduce the notion of *concept completion*, which is used to manipulate an interpretation $\mathcal{I}$ so that $\mathcal{I}_{\mathcal{R}}$ is a model of $\mathcal{H}$.

Definition 28 (Concept Completion) *Let $\mathcal{H}$ be a hypothesis, $\mathcal{R}$ be a finite set of RIs, and let $\mathcal{I}$ be an interpretation. Define a sequence of interpretations $\mathcal{I}_0, \mathcal{I}_1, \ldots$ by setting*

- *$\mathcal{I}^0 = \mathcal{I}$;*
- *assume $\mathcal{I}^n$ has been defined. If there is no $k \leq n$ such that there is a $d \in \Delta^{\mathcal{I}^0}$ with $d \in D^{\mathcal{I}^k_{\mathcal{R}}}$, $D \sqsubseteq A \in \mathcal{H}$ but $d \notin A^{\mathcal{I}^k}$, then let $\mathcal{I}^{n+1} := \mathcal{I}_n$. Otherwise assume that k is minimal with this property and define $\mathcal{I}^{n+1}$ as $\mathcal{I}^n$ except that $A^{\mathcal{I}^{n+1}} := A^{\mathcal{I}^n} \cup \{d\}$.*

The concept completion $\mathcal{I}^{(\mathcal{R})}_{\mathcal{H}}$ of $\mathcal{I}$ w.r.t. $\mathcal{H}$ for $\mathcal{R}$ is defined as the limit $\bigcup_{n \geq 0} \mathcal{I}^n$.

Note that the concept completion $\mathcal{I}^{(\mathcal{R})}_{\mathcal{H}}$ of $\mathcal{I}$ w.r.t. $\mathcal{H}$ for $\mathcal{R}$ modifies $\mathcal{I}$ only by adding elements to the extension of concept names; there is no role completion in this case (we use $\mathcal{R}$ only to check if $d \in D^{\mathcal{I}^k_{\mathcal{R}}}$). By the definition of $\mathcal{J} = \mathcal{I}^{(\mathcal{R})}_{\mathcal{H}}$, we have that $\mathcal{J}_{\mathcal{R}} \models \mathcal{H}$. Since $\mathcal{H}_{add}$ contains only $\mathcal{EL}_{\mathsf{lhs}}$ concept inclusions, if $\mathcal{J}_{\mathcal{R}} \models \mathcal{H}$ then $\mathcal{J} \models \mathcal{H}_{add}$.

We write $\mathcal{I}^{d\downarrow}$ to denote the multi-edge subtree rooted in d. Recall that we use $\mathcal{I}^{-d\downarrow}$ to denote $\mathcal{I}$ with the multi-edge subtree rooted in d removed.

Lemma 29 *Given a positive counterexample $C \sqsubseteq A$ for $\mathcal{T}$ relative to $\mathcal{H}$, one can construct a single-edge tree shaped $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel $\mathcal{I}$ for $\mathcal{H}$ in polynomial time in $|\mathcal{T}| + |C|$ by making membership queries.*

Proof. Let $C \sqsubseteq A$ be a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}$. Let $\mathcal{I}$ be the single-edge tree interpretation of C. First observe that $\mathcal{I}_{\mathcal{R}} \not\models \mathcal{T}$: since $\mathcal{H} \not\models C \sqsubseteq A$, we know that A does not occur as a top-level conjunct in C. Consequently, $\rho_{\mathcal{I}_{\mathcal{R}}} \in (C \setminus A)^{\mathcal{I}_{\mathcal{R}}}$ and thus $\mathcal{I}_{\mathcal{R}} \not\models \mathcal{T}$. $\mathcal{J}$ is constructed by applying the following rules to $\mathcal{J} := \mathcal{I}$.

1. Let $\mathcal{J}^{(\mathcal{R})}_{\mathcal{H}}$ be the concept completion of $\mathcal{J}$ w.r.t. $\mathcal{H}$ for $\mathcal{R}$. Set $\mathcal{J} := \mathcal{J}^{(\mathcal{R})}_{\mathcal{H}}$. That is, transform $\mathcal{J}$ into $\mathcal{J}^{(\mathcal{R})}_{\mathcal{H}}$ by exhaustively applying the CIs in $\mathcal{H}$ as rules: if $D \sqsubseteq B \in \mathcal{H}$ and $d \in D^{\mathcal{J}_{\mathcal{R}}}$, then add d to $B^{\mathcal{J}}$.
2. Replace $\mathcal{J}$ by a minimal single-edge subtree of $\mathcal{J}$ refuting $\mathcal{T}$ to address Condition 1 of $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$: replace $\mathcal{J}$ by $\mathcal{J}^{d\downarrow}$ if $\mathcal{J}^{d\downarrow}$ is minimal with $\mathcal{J}^{d\downarrow}_{\mathcal{R}} \not\models \mathcal{T}$ (checked by making queries of the form $C_{\mathcal{J}^{d\downarrow}} \sqsubseteq A$).
3. Exhaustively remove subtrees from $\mathcal{J}$ until Condition 2 of $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$ - countermodels for $\mathcal{H}$ is also satisfied: if $(\mathcal{J}^{-d\downarrow})_{\mathcal{R}} \not\models \mathcal{T}$ (checked by making queries of the form $C_{\mathcal{J}^{-d\downarrow}} \sqsubseteq A$), then replace $\mathcal{J}$ by $\mathcal{J}^{-d\downarrow}$.

Now we show that the interpretation $\mathcal{J}$ constructed above has the required properties:

- $\mathcal{J}_{\mathcal{R}} \models \mathcal{H}$: clearly, the interpretation $\mathcal{J}_{\mathcal{R}}$ constructed in Step 1 is a model of $\mathcal{H}$. Taking subtrees and removing subtrees from $\mathcal{J}$ preserves being a model of $\mathcal{H}$, and so $\mathcal{J}_{\mathcal{R}} \models \mathcal{H}$.
- $\mathcal{J}_{\mathcal{R}} \not\models \mathcal{T}$: the interpretation $\mathcal{J}$ constructed in Step 1 (and its role completion $\mathcal{J}_{\mathcal{R}}$) is not a model of $\mathcal{T}$. In fact, we can use $C_{\mathcal{J}} \sqsubseteq A$ as a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}$ instead of $C \sqsubseteq A$. Observe that $\emptyset \models C_{\mathcal{J}} \sqsubseteq C$, and thus $\mathcal{T} \models C \sqsubseteq A$ implies $\mathcal{T} \models C_{\mathcal{J}} \sqsubseteq A$. On the other hand, since $\mathcal{H} \not\models C \sqsubseteq A$, we have $\rho_{\mathcal{J}} \notin A^{\mathcal{J}}$. Thus $\mathcal{J}_{\mathcal{R}} \not\models \mathcal{T}$. Steps 2 and 3 preserve the condition that $\mathcal{J}_{\mathcal{R}}$ is not a model of $\mathcal{T}$ and so $\mathcal{J}_{\mathcal{R}} \not\models \mathcal{T}$.
- $\mathcal{J}$ satisfies Condition 1 for $\mathcal{T}$-essential models because of Step 2.
- $\mathcal{J}$ satisfies Condition 2 for $\mathcal{T}$-essential models because of Step 3.

❏

Lemma 30 *If $\mathcal{I}$ is a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$, then $|\Delta^{\mathcal{I}}| \leq |\mathcal{T}|$.*

Proof. Let $\mathcal{I}$ be a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$. As $\mathcal{I}$ is an $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$, $\mathcal{I}_{\mathcal{R}} \not\models \mathcal{T}$. By Property 2 of being a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$ we have $(\mathcal{I}^{-\rho_{\mathcal{I}}})_{\mathcal{R}} \models \mathcal{T}$. It follows that there is a $C \sqsubseteq A \in \mathcal{T}$ such that $\rho_{\mathcal{I}_{\mathcal{R}}} \in (C \setminus A)^{\mathcal{I}_{\mathcal{R}}}$. Consequently, there is a simulation $\mathcal{I}_C \Rightarrow \mathcal{I}_{\mathcal{R}}$, where $\mathcal{I}_C$ is the single-edge tree interpretation of C. Since $\mathcal{I}_C$ is single-edge tree shaped, we can assume that the simulation $S \subseteq \Delta^{\mathcal{I}_C} \times \Delta^{\mathcal{I}_{\mathcal{R}}}$ is a total function (that is, a homomorphism). To show that $|\Delta^{\mathcal{I}_{\mathcal{R}}}| = |\Delta^{\mathcal{I}}| \leq |C|$ and thus $|\Delta^{\mathcal{I}}| \leq |\mathcal{T}|$ as required, it clearly suffices to show that S is surjective. Assume that this is not the case, and let $d \in \Delta^{\mathcal{I}}$ be outside the range of S. Let $\mathcal{J} = \mathcal{I}^{-d\downarrow}$ (Property 1 $\mathcal{I}$ is single-edge tree shaped). All descendants of d must be outside the range of S as well and thus S is a simulation $\mathcal{I}_C \Rightarrow \mathcal{J}_{\mathcal{R}}$. Therefore, $\rho_{\mathcal{J}_{\mathcal{R}}} \in C^{\mathcal{J}_{\mathcal{R}}}$, which implies $\mathcal{J}_{\mathcal{R}} \not\models C \sqsubseteq A \in \mathcal{T}$, in contradiction to $\mathcal{I}$ being a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$: Property 3 requires $\mathcal{J}_{\mathcal{R}} \models \mathcal{T}$. ❏

For two $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels $\mathcal{I}$ and $\mathcal{J}$ for $\mathcal{H}$ (with roots $\rho_{\mathcal{I}}$ and $\rho_{\mathcal{J}}$ respectively), let $\mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}_{\mathcal{R}}$ be the maximal multi-edge reduced tree interpretation that is contained in $\mathcal{I}_{\mathcal{R}} \times \mathcal{J}_{\mathcal{R}}$ with root $(\rho_{\mathcal{I}}, \rho_{\mathcal{J}})$. In Line 10 of Algorithm 6, we compute $\mathcal{J}' \subseteq \mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}_{\mathcal{R}}$, where $\mathcal{J}'$ is a subinterpretation of $\mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}_{\mathcal{R}}$ that is obtained from $\mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}_{\mathcal{R}}$ by removing subtrees.

Lemma 31 *Let $\mathcal{I}$ and $\mathcal{J}$ be $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ such that $\mathcal{I}_{\mathcal{R}} \times_{\rho} \mathcal{J}_{\mathcal{R}} \not\models \mathcal{T}$. Let $\mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}_{\mathcal{R}}$ be the result of multi-edge reducing $\mathcal{I}_{\mathcal{R}} \times_{\rho} \mathcal{J}_{\mathcal{R}}$. If $\mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}_{\mathcal{R}}$ is single-edge tree shaped then one can construct a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel $\mathcal{J}' \subseteq \mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}_{\mathcal{R}}$ for $\mathcal{H}$ in polynomial time in $|\mathcal{T}|$ by making membership queries.*

Proof. Let $\mathcal{I}$ and $\mathcal{J}$ be $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ with $\mathcal{I}_{\mathcal{R}} \times_{\rho} \mathcal{J}_{\mathcal{R}} \not\models \mathcal{T}$.

Set $\mathcal{J}' = \mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}_{\mathcal{R}}$ and then exhaustively remove subtrees from $\mathcal{J}'$ until Condition 2 of $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ is satisfied: if $(\mathcal{J}'^{-d\downarrow})_{\mathcal{R}} \not\models \mathcal{T}$ (checked by making membership queries, then replace $\mathcal{J}'$ by $\mathcal{J}'^{-d\downarrow}$. Clearly, the new $\mathcal{J}'$ is an $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$. Moreover, it is $\mathcal{T}$-essential:

- $(\mathcal{J}'^{-\rho_{\mathcal{J}'}})_\mathcal{R} \models \mathcal{T}$: by Lemma 18, $(\mathcal{I}^{-\rho_\mathcal{I}})_\mathcal{R} \models \mathcal{T}$ and $(\mathcal{J}^{-\rho_\mathcal{J}})_\mathcal{R} \models \mathcal{T}$ imply $(\mathcal{I}^{-\rho_\mathcal{I}})_\mathcal{R} \times (\mathcal{J}^{-\rho_\mathcal{J}})_\mathcal{R} \models \mathcal{T}$. It follows from Lemma 24, that $(\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R})_\mathcal{R} \models \mathcal{T}$ iff $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R} \models \mathcal{T}$. Also, if $(\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R})_\mathcal{R} \models \mathcal{T}$ then $\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R} \models \mathcal{T}$. Now $\mathcal{J}'^{-\rho_{\mathcal{J}'}}$ can be obtained from $\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R}^{-\rho_{\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R}}}$ by removing subtrees, and removing subtrees clearly preserves being a model of an $\mathcal{ELH}_\mathsf{lhs}$ TBox.
- $(\mathcal{J}'^{-d\downarrow})_\mathcal{R} \models \mathcal{T}$ for all $d \in \Delta^{\mathcal{J}'} \setminus \{\rho_{\mathcal{J}'}\}$: otherwise, the subtree rooted at d would have been removed during the construction of $\mathcal{J}'$.

So we have seen that one can construct a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel $\mathcal{J}' \subseteq \mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R}$ for $\mathcal{H}$ using only polynomially many membership queries in $|\mathcal{T}| + |\mathcal{I}| + |\mathcal{J}|$. Since $\mathcal{I}$ and $\mathcal{J}$ are $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$, by Lemma 30, $|\Delta^\mathcal{I}| \le |\mathcal{T}|$ and $|\Delta^\mathcal{J}| \le |\mathcal{T}|$. Then we have that $\mathcal{J}'$ is constructed in polynomial time in $|\mathcal{T}|$ by making membership queries. ❑

$\mathcal{I}$-candidates for $\mathcal{J}$

We want to show that single-edge tree shaped $\mathcal{I}$-candidates for $\mathcal{J}$ are polynomially computable (Lemma 36). First we introduce some notions used in our proofs.

Definition 32 (Bisimulation) *Let $\mathcal{I}, \mathcal{J}$ be interpretations, $d_0 \in \Delta^\mathcal{I}$ and $e_0 \in \Delta^\mathcal{J}$. A relation $S \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{J}$ is a* bisimulation *between $(\mathcal{I}, d_0)$ and $(\mathcal{J}, e_0)$ if $(d_0, e_0) \in S$ and the following conditions are satisfied:*

- *for all concept names $A \in \mathsf{N_C}$ and all $(d, e) \in S$, $d \in A^\mathcal{I}$ iff $e \in A^\mathcal{J}$;*
- *for all role names $r \in \mathsf{N_R}$, all $(d, e) \in S$ and all $d' \in \Delta^\mathcal{I}$, if $(d, d') \in r^\mathcal{I}$ then there exists $e' \in \Delta^\mathcal{J}$ such that $(e, e') \in r^\mathcal{J}$ and $(d', e') \in S$;*
- *for all role names $r \in \mathsf{N_R}$, all $(d, e) \in S$ and all $e' \in \Delta^\mathcal{J}$, if $(e, e') \in r^\mathcal{J}$ then there exists $d' \in \Delta^\mathcal{I}$ such that $(d, d') \in r^\mathcal{I}$ and $(d', e') \in S$.*

We write $(\mathcal{I}, d_0) \Leftrightarrow (\mathcal{J}, e_0)$ when we have a bisimulation between $(\mathcal{I}, d_0)$ and $(\mathcal{J}, e_0)$. If $\mathcal{I}$ and $\mathcal{J}$ are tree shaped then we write $\mathcal{I} \Leftrightarrow \mathcal{J}$ as a shorthand for $(\mathcal{I}, \rho_\mathcal{I}) \Leftrightarrow (\mathcal{J}, \rho_\mathcal{J})$, where $\rho_\mathcal{I}$ and $\rho_\mathcal{J}$ are the roots of $\mathcal{I}$ and $\mathcal{J}$, respectively.

One can obtain a single-edge tree shaped interpretation from a multi-edge one by *unfolding* it. For a multi-edge tree shaped interpretation $\mathcal{I}$ rooted in $\rho_\mathcal{I}$, let $\mathsf{paths}(\mathcal{I}) = \{d_0 \cdot r_0 \cdot \ldots \cdot r_{n-1} \cdot d_n \mid d_0, \ldots, d_n \in \Delta^\mathcal{I}, r_0, \ldots, r_{n-1} \in \mathsf{N_R}$, $d_0 = \rho_\mathcal{I}$ and $(d_i, d_{i+1}) \in r_i^\mathcal{I}, 0 \le i < n\}$ be the set of paths in $\mathcal{I}$. We also define $\mathsf{tail}_\mathcal{I}(p) = d_n$ as the last element in p.

Definition 33 (Unfolding) *The unfolding $\mathcal{J}$ of a multi-edge tree shaped interpretation $\mathcal{I}$ is defined as:*

- *$\Delta^\mathcal{J} = \mathsf{paths}(\mathcal{I})$;*
- *$A^\mathcal{J} = \{p \in \mathsf{paths}(\mathcal{I}) \mid \mathsf{tail}_\mathcal{I}(p) \in A^\mathcal{I}\}$;*
- *$r^\mathcal{J} = \{(p, p') \in \mathsf{paths}(\mathcal{I}) \times \mathsf{paths}(\mathcal{I}) \mid p' = p \cdot r \cdot d\}$.*

We observe that given an interpretation $\mathcal{I}$, we have that

$$S = \{(\mathsf{tail}_\mathcal{I}(p), p) \mid p \in \mathsf{paths}(\mathcal{I})\}$$

is a bisimulation between $\mathcal{I}$ and $\mathcal{J}$ (Blackburn, Benthem, and Wolter 2006):

Lemma 34 *Let $\mathcal{J}$ be the result of unfolding a multi-edge tree shaped interpretation $\mathcal{I}$. Then, $\mathcal{I} \Leftrightarrow \mathcal{J}$.*

For a multi-edge interpretation $\mathcal{I}$, let $[(d, e)]_\mathcal{I}$ denote the set of role names r such that $(d, e) \in r^\mathcal{I}$, where $d, e \in \Delta^\mathcal{I}$. We say that a pair of elements is multi-edge if $|[(d, e)]_\mathcal{I}| > 1$. If $|[(d, e)]_\mathcal{I}| = 1$ then we say that (d, e) is single-edge. If (d, e) is a multi-edge pair then we say that d, e are members of a multi-edge pair. The predecessors of d are elements of the set $\mathsf{nodes}(p) \setminus \{d\}$ where p is a path such that $\mathsf{tail}(p) = d$. The successors of e are the elements which have e as a predecessor.

Let $\mathcal{I}$,$\mathcal{J}$ be $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels such that $\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R}$ is not single-edge tree shaped. Let $\mathcal{I}^\times$ be an $\mathcal{I}$-candidate for $\mathcal{J}$. By definition of $\mathcal{I}^\times$, there is a homomorphism $h : \mathcal{I}^\times_\mathcal{R} \to \mathcal{I}_\mathcal{R}$ and a homomorphism $g : \mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R} \to \mathcal{I}^\times_\mathcal{R}$ (mapping the respective roots). We describe an operation that transforms a multi-edge pair (d, e) in $\mathcal{I}^\times$ into a single-edge one by replacing it by the unique role s such that $(h(d), h(e)) \in s^\mathcal{I}$.

Definition 35 *We restrict to $\mathcal{I}$ a multi-edge pair (d, e) in an $\mathcal{I}$-candidate $\mathcal{I}^\times$ for $\mathcal{J}$ as follows. For all $r \in [(d, e)]_{\mathcal{I}^\times}$, remove (d, e) from $r^{\mathcal{I}^\times}$ Let s be the unique role connecting the pair $(h(d), h(e))$ in $\mathcal{I}$. That is, s such that $(h(d), h(e)) \in s^\mathcal{I}$. Now, add (d, e) to $s^{\mathcal{I}^\times}$.*

Note that this operation preserves the homomorphisms h and g of an $\mathcal{I}$-candidate for $\mathcal{J}$. We can now show how we construct single-edge tree shaped $\mathcal{I}$-candidates for $\mathcal{J}$.

Lemma 36 *Given two $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels $\mathcal{I}$ and $\mathcal{J}$ such that $\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R}$ is not single-edge tree shaped, one can compute in polynomial time in $|\mathcal{T}|$ a single-edge tree shaped $\mathcal{I}$-candidate $\mathcal{I}^\times$ for $\mathcal{J}$ with $|\Delta^{\mathcal{I}^\times}|$ bounded by $|\mathcal{T}|^2 \cdot (|\mathcal{R}| + 1)$.*

Proof. We show how to construct in polynomial time a single-edge tree shaped $\mathcal{I}$-candidate $\mathcal{I}^\times$ for $\mathcal{J}$. Initially set $\mathcal{I}^\times := (\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R})$. By definition of $\mathcal{I}^\times$: there is a homomorphism $h : \mathcal{I}^\times_\mathcal{R} \to \mathcal{I}_\mathcal{R}$ (defined by the projection of $\mathcal{I}_\mathcal{R} \times^\mathsf{r}_\rho \mathcal{J}_\mathcal{R}$ to $\mathcal{I}_\mathcal{R}$); and an automorphism $g : \mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}_\mathcal{R} \to \mathcal{I}^\times_\mathcal{R}$. By assumption $\mathcal{I}^\times$ has a multi-edge pair. Now, apply the following steps.

1. Exhaustively apply the following rule: if $\mathcal{I}^\times$ has a multi-edge pair and $\mathcal{I}_\mathcal{R} \not\Rightarrow \mathcal{J}'_\mathcal{R}$, where $\mathcal{J}'$ is the result of restricting to $\mathcal{I}$ a multi-edge pair in $\mathcal{I}^\times$ (Definition 35) and multi-edge reducing; then replace $\mathcal{I}^\times$ by $\mathcal{J}'$
2. Let $\mathcal{I}^{\times,(\mathcal{R})}_\mathcal{H}$ be the concept completion of $\mathcal{I}^\times$ w.r.t. $\mathcal{H}$ for $\mathcal{R}$ (Definition 28). Set $\mathcal{I}^\times := \mathcal{I}^{\times,(\mathcal{R})}_\mathcal{H}$. That is, transform $\mathcal{I}^\times$ into $\mathcal{I}^{\times,(\mathcal{R})}_\mathcal{H}$ by exhaustively applying the CIs in $\mathcal{H}$ as rules: if $D \sqsubseteq A' \in \mathcal{H}$ and $d \in D^{\mathcal{I}^\times_\mathcal{R}}$, then add d to $A'^{\mathcal{I}^\times}$.
3. Unfold $\mathcal{I}^\times$ (Definition 33).

Before we show that $\mathcal{I}^\times$ is a single-edge tree shaped $\mathcal{I}$-candidate for $\mathcal{J}$ with $|\Delta^{\mathcal{I}^\times}|$ bounded by $|\mathcal{T}|^2 \cdot (|\mathcal{R}| + 1)$, we show the following 3 claims. Claim 1 is immediate.

Claim 1. Let $\mathcal{I}$ and $\mathcal{J}$ be two multi-edge tree interpretations. Assume $S \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{J}$ is a simulation $\mathcal{I} \Rightarrow \mathcal{J}$.

For all $(d,e) \in S$, there is a simulation $\mathcal{I}^{d\downarrow} \Rightarrow \mathcal{J}^{e\downarrow}$, where $\mathcal{I}^{d\downarrow}$ and $\mathcal{J}^{e\downarrow}$ denote the multi-edge subtree of $\mathcal{I}$ and $\mathcal{J}$ rooted in d and e, respectively.

Claim 2. Let $\mathcal{I}$ and $\mathcal{J}$ be multi-edge reduced tree interpretations. If $S \subseteq \Delta^{\mathcal{I}_\mathcal{R}} \times \Delta^{\mathcal{J}_\mathcal{R}}$ is a simulation $\mathcal{I}_\mathcal{R} \Rightarrow \mathcal{J}_\mathcal{R}$ and $Q \subseteq \Delta^{\mathcal{J}_\mathcal{R}} \times \Delta^{\mathcal{I}_\mathcal{R}}$ is a simulation $\mathcal{J}_\mathcal{R} \Rightarrow \mathcal{I}_\mathcal{R}$ then there is a bisimulation $S' \subseteq S$ between $\mathcal{I}_\mathcal{R}$ and $\mathcal{J}_\mathcal{R}$. That is, $\mathcal{I}_\mathcal{R} \Leftrightarrow \mathcal{J}_\mathcal{R}$.

Define $S' \subseteq S$ as follows: for all $(d,e) \in \Delta^{\mathcal{I}_\mathcal{R}} \times \Delta^{\mathcal{J}_\mathcal{R}}$, $(d,e) \in S'$ iff $(e,d) \in Q$ and $(d,e) \in S$. Now, we show that S' is a bisimulation between $\mathcal{I}_\mathcal{R}$ and $\mathcal{J}_\mathcal{R}$. That is,

1. for all concept names $A \in \mathsf{N_C}$ and all $(d,e) \in S'$, $d \in A^{\mathcal{I}_\mathcal{R}}$ iff $e \in A^{\mathcal{J}_\mathcal{R}}$;
2. for all role names $r \in \mathsf{N_R}$, all $(d,e) \in S'$ and all $d' \in \Delta^{\mathcal{I}_\mathcal{R}}$, if $(d,d') \in r^{\mathcal{I}_\mathcal{R}}$ then there exists $e' \in \Delta^{\mathcal{J}_\mathcal{R}}$ such that $(e,e') \in r^{\mathcal{J}_\mathcal{R}}$ and $(d',e') \in S'$;
3. for all role names $r \in \mathsf{N_R}$, all $(d,e) \in S'$ and all $e' \in \Delta^{\mathcal{J}_\mathcal{R}}$, if $(e,e') \in r^{\mathcal{J}_\mathcal{R}}$ then there exists $d' \in \Delta^{\mathcal{I}_\mathcal{R}}$ such that $(d,d') \in r^{\mathcal{I}_\mathcal{R}}$ and $(d',e') \in S'$.

Point 1 is clear. We show in Claims 1 and 2 that Points 2 and 3 hold for roles in $\mathcal{I}$ and $\mathcal{J}$. For the roles included in $\mathcal{I}_\mathcal{R}$ by the role completion of $\mathcal{I}$ with $\mathcal{R}$, we have that if $(d,e) \in S'$ and $(d,d') \in s^{\mathcal{I}_\mathcal{R}}$, where $R \models r \sqsubseteq s$ for some $r \in [(d,d')]^{\mathcal{I}}$, then by Claim 1 there is $e' \in \Delta^{\mathcal{J}_\mathcal{R}}$ such that $(e,e') \in r^{\mathcal{J}_\mathcal{R}}$ and $(d',e') \in S'$. Since $R \models r \sqsubseteq s$ and $(e,e') \in r^{\mathcal{J}_\mathcal{R}}$, we have that $(e,e') \in s^{\mathcal{J}_\mathcal{R}}$. The same argument holds for roles included in $\mathcal{J}_\mathcal{R}$ by the role completion of $\mathcal{J}$ with $\mathcal{R}$.

Claim 2.1. For all role names $r \in \mathsf{N_R}$, all $(d,e) \in S'$ and all $d' \in \Delta^{\mathcal{I}} = \Delta^{\mathcal{I}_\mathcal{R}}$, if $(d,d') \in r^{\mathcal{I}}$ then there exists $e' \in \Delta^{\mathcal{J}_\mathcal{R}}$ such that $(e,e') \in r^{\mathcal{J}_\mathcal{R}}$ and $(d',e') \in S'$.

If $(d,e) \in S'$ then $(d,e) \in S$ and $(e,d) \in Q$. If $(d,e) \in S$ and $(d,d') \in r^{\mathcal{I}}$ then there is $e' \in \Delta^{\mathcal{J}_\mathcal{R}}$ such that $(e,e') \in r^{\mathcal{J}_\mathcal{R}}$ and $(d',e') \in S$. If $(e',d') \in Q$ then we are done. Suppose this is not the case. Then there is $d'' \neq d' \in \Delta^{\mathcal{I}_\mathcal{R}}$ such that $(d,d'') \in r^{\mathcal{I}_\mathcal{R}}$ and $(e',d'') \in Q$. Then, by Claim 1, $\mathcal{I}_\mathcal{R}^{d'\downarrow} \Rightarrow \mathcal{J}_\mathcal{R}^{e'\downarrow}$ and $\mathcal{J}_\mathcal{R}^{e'\downarrow} \Rightarrow \mathcal{I}_\mathcal{R}^{d''\downarrow}$, which means that $\mathcal{I}_\mathcal{R}^{d'\downarrow} \Rightarrow \mathcal{I}_\mathcal{R}^{d''\downarrow}$. Let $\mathcal{I}^{-r(d,d')}$ be the result of removing (d,d') from $r^{\mathcal{I}}$. Since $(d,d'') \in r^{\mathcal{I}_\mathcal{R}}$ and $\mathcal{I}_\mathcal{R}^{d'\downarrow} \Rightarrow \mathcal{I}_\mathcal{R}^{d''\downarrow}$, we have that $\mathcal{I}_\mathcal{R} \Rightarrow (\mathcal{I}^{-r(d,d')})_\mathcal{R}$. This contradicts the fact that $\mathcal{I}$ is multi-edge reduced.

Claim 2.2. For all role names $r \in \mathsf{N_R}$, all $(d,e) \in S'$ and all $e' \in \Delta^{\mathcal{J}} = \Delta^{\mathcal{J}_\mathcal{R}}$, if $(e,e') \in r^{\mathcal{J}}$ then there exists $d' \in \Delta^{\mathcal{I}_\mathcal{R}}$ such that $(d,d') \in r^{\mathcal{I}_\mathcal{R}}$ and $(d',e') \in S'$.

The argument is analogous to Claim 2.1.

Claim 3. Let $\mathcal{I}$ be a single-edge tree interpretation and $\mathcal{J}$ a multi-edge tree interpretation with a multi-edge pair, where both $\mathcal{I}$ and $\mathcal{J}$ are multi-edge reduced. If there is a homomorphism $h : \mathcal{J}_\mathcal{R} \to \mathcal{I}_\mathcal{R}$ mapping the roots of $\mathcal{I}$ and $\mathcal{J}$ then $\mathcal{I}_\mathcal{R} \not\Rightarrow \mathcal{J}_\mathcal{R}$.

Suppose to the contrary that $\mathcal{I}_\mathcal{R} \Rightarrow \mathcal{J}_\mathcal{R}$. By assumption $\mathcal{I}$ and $\mathcal{J}$ are multi-edge reduced and there is a homomorphism $h : \mathcal{J}_\mathcal{R} \to \mathcal{I}_\mathcal{R}$ mapping the roots. Then we can apply Claim 2 to obtain that there is a bisimulation $S \subseteq h$ between $\mathcal{I}_\mathcal{R}$ and $\mathcal{J}_\mathcal{R}$. By assumption there is a multi-edge pair (d,d') in $\mathcal{J}$. Since h is a homomorphism and $S \subseteq h$, we have that there is a unique pair (e,e') in $\mathcal{I}$ such that $(d,e),(d',e') \in S$. As $\mathcal{I}$ is single-edge tree shaped, there is a unique role s connecting (e,e') in $\mathcal{I}$. By definition of $\mathcal{I}_\mathcal{R}$, $(e,e') \in s^{\mathcal{I}_\mathcal{R}}$ and for all $r \in [(e,e')]_{\mathcal{I}_\mathcal{R}}$, $\mathcal{R} \models s \sqsubseteq r$. Since (d,d') is multi-edge in $\mathcal{J}$, we have that $s \notin [(d,d')]_\mathcal{J}$ (otherwise we would have redundant role assertions in $\mathcal{J}$). As $\mathcal{I}_\mathcal{R} \Leftrightarrow \mathcal{J}_\mathcal{R}$, there is $d'' \in \Delta^{\mathcal{J}_\mathcal{R}}$ such that $(d,d'') \in s^{\mathcal{J}_\mathcal{R}}$ and $(d'',e') \in S$. By Claim 1, $\mathcal{J}_\mathcal{R}^{d'\downarrow} \Rightarrow \mathcal{I}_\mathcal{R}^{e'\downarrow}$ and $\mathcal{I}_\mathcal{R}^{e'\downarrow} \Rightarrow \mathcal{J}_\mathcal{R}^{d''\downarrow}$. Then we have that $\mathcal{J}_\mathcal{R}^{d'\downarrow} \Rightarrow \mathcal{J}_\mathcal{R}^{d''\downarrow}$. Now, let $\mathcal{J}^{-d'\downarrow}$ be the result of removing the multi-edge subtree rooted in d' from $\mathcal{J}$. By definition of $\mathcal{J}^{-d'\downarrow}$, we have that $\mathcal{J}_\mathcal{R} \Rightarrow (\mathcal{J}^{-d'\downarrow})_\mathcal{R}$. This contradicts the fact that $\mathcal{J}$ is multi-edge reduced. Then, $\mathcal{I}_\mathcal{R} \not\Rightarrow \mathcal{J}_\mathcal{R}$. This finishes the proof of Claim 3.

Now we show that the interpretation $\mathcal{I}^\times$ constructed above is a single-edge tree shaped $\mathcal{I}$-candidate for $\mathcal{J}$.

- The fact that there are homomorphisms $h : \mathcal{I}^\times_\mathcal{R} \to \mathcal{I}_\mathcal{R}$ and $g : \mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R} \to \mathcal{I}^\times_\mathcal{R}$ mapping the respective roots is justified by the definition of $\mathcal{I}^\times$ and the rule of restricting to $\mathcal{I}$ multi-edge pairs, which maintain these homomorphisms.
- Since $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R}$ is not single-edge tree shaped and $\mathcal{I}$ is single-edge tree shaped we have that $(\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R}) \not\Rightarrow \mathcal{I}_\mathcal{R}$ (Claim 3). Then initially we have that $\mathcal{I}_\mathcal{R} \not\Rightarrow \mathcal{I}^\times_\mathcal{R}$. As we check if this property is maintained before restricting to $\mathcal{I}$ multi-edge pairs, we have that at all times $\mathcal{I}_\mathcal{R} \not\Rightarrow \mathcal{I}^\times_\mathcal{R}$. Also, by assumption, initially $\mathcal{I}^\times$ is reduced and has a multi-edge pair. So we know that at least one multi-edge pair in $\mathcal{I}^\times$ is restricted to $\mathcal{I}$. After we restrict this multi-edge pair to $\mathcal{I}$, we have that $\mathcal{I}^\times_\mathcal{R} \not\Rightarrow (\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R})$. This property is maintained by the other steps.
- Step 2 ensures that $\mathcal{I}^\times_\mathcal{R} \models \mathcal{H}$. Since $\mathcal{I}_\mathcal{R} \models \mathcal{H}$ and $(\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}_\mathcal{R})_\mathcal{R} \models \mathcal{H}$, Step 2 preserves the homomorphisms and simulations above.
- The unfolding in Step 3 ensures that $\mathcal{I}^\times$ is single-edge tree shaped.

It remains to show that the unfolding of $\mathcal{I}^\times$ in Step 3 above has size polynomial in $|\mathcal{T}|$. This is a consequence of the following claim.

Claim 4. For each multi-edge pair (d,e) in $\mathcal{I}^\times$ the successors of e are not members of multi-edge pairs.

Suppose the lemma does not hold. Then, let (d,e) and (d',e') be two multi-edge pairs in $\mathcal{I}^\times$ with d',e' as successors of e. Now, let $\mathcal{I}'$ be the result of restricting $\mathcal{I}^\times$ to $\mathcal{I}$ one of the multi-edge pairs and multi-edge reducing. Let (d,e) be the restricted multi-edge pair (the case where (d',e') is the restricted pair is similar). Since $\mathcal{I}^\times$ was multi-edge reduced and the path containing (d, e) is exhaustively restricted (Step 1 above), (d', e') is a multi-edge pair in $\mathcal{I}'$. By definition of $\mathcal{I}'_\mathcal{R}$ there is a homomorphism $g' : \mathcal{I}'_\mathcal{R} \to \mathcal{I}_\mathcal{R}$. Then, we can apply Claim 3 to obtain that $\mathcal{I}_\mathcal{R} \not\Rightarrow \mathcal{I}'_\mathcal{R}$. Now this contradicts the fact that $\mathcal{I}^\times$ was exhaustively restricted in Step 1. Then, for each multi-edge pair (d, e) in $\mathcal{I}'$ the successors of e are not members of multi-edge pairs.

By Claim 4, the successors of multi-edge pairs are not members of multi-edge pairs. So the unfolding of $\mathcal{I}^\times$ increases the number of nodes by at most $|\mathcal{R}| \cdot |\Delta^{\mathcal{I}^\times}|$. By assumption $\mathcal{I}$ and $\mathcal{J}$ are $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$, then, by Lemma 30, $|\Delta^\mathcal{I}| \leq |\mathcal{T}|$ and $|\Delta^\mathcal{J}| \leq |\mathcal{T}|$. This means that $|\Delta^{\mathcal{I}^\times}| \leq |\mathcal{T}|^2$. Since $|\Delta^{\mathcal{I}^\times}| \leq |\mathcal{T}|^2$ we have that $|\Delta^{\mathcal{I}^{\times\prime}}| \leq |\mathcal{R}| \cdot |\mathcal{T}|^2 + |\mathcal{T}|^2 = |\mathcal{T}|^2 \cdot (|\mathcal{R}| + 1)$. ❑

Polynomial Time Bound on the Algorithm

By Lemma 36, in Line 12 of Algorithm 7, $\mathcal{I}^\times$ is a polynomial single-edge tree shaped interpretation. Next, Algorithm 6 recursively calls the function "RefineCounterModel". Lemma 38 shows that the number of recursive calls is polynomial with respect to $|\mathcal{T}|$. To prove Lemma 38, we first show a technical lemma.

Lemma 37 *Let $\mathcal{I}^0, \mathcal{I}^1, \ldots, \mathcal{I}^n$ be a sequence of $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ such that, for all $0 \leq l \leq n$, $\mathcal{I}^l_\mathcal{R} \not\Rightarrow \mathcal{I}^{l+1}_\mathcal{R}$ and $\mathcal{I}^{l+1}_\mathcal{R} \Rightarrow \mathcal{I}^l_\mathcal{R}$. Then, $n \leq |\mathcal{T}|(|\Sigma_\mathcal{T}| + |\mathcal{R}| + 1)$.*

Proof. We first show that for every $0 \leq l \leq n$ either

1. there is a concept name A such that $\rho_{\mathcal{I}^l} \in A^{\mathcal{I}^l}$ and $\rho_{\mathcal{I}^{l+1}} \notin A^{\mathcal{I}^{l+1}}$ or
2. $\mathcal{I}^{l+1}_\mathcal{R} \Rightarrow \mathcal{I}^l_\mathcal{R}$ via a surjective simulation.

For a proof by contradiction assume that there is $0 \leq l \leq n$ such that neither Point 1 nor Point 2 holds. By assumption $\mathcal{I}^{l+1}_\mathcal{R} \Rightarrow \mathcal{I}^l_\mathcal{R}$. Since $\mathcal{I}^{l+1}$ is a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$, there is a $C \sqsubseteq A \in \mathcal{T}$ such that $\mathcal{I}^{l+1}_\mathcal{R} \not\models^\rho C \sqsubseteq A$. Let $\mathcal{J}$ be the subinterpretation of $\mathcal{I}^l_\mathcal{R}$ determined by the range of the simulation $S \subseteq \Delta^{\mathcal{I}^{l+1}_\mathcal{R}} \times \Delta^{\mathcal{I}^l_\mathcal{R}}$ from $\mathcal{I}^{l+1}_\mathcal{R}$ to $\mathcal{I}^l_\mathcal{R}$. Then $\rho_\mathcal{J} \in C^\mathcal{J}$ and so, since $\rho_\mathcal{J} \notin A^\mathcal{J}$ because Point 1 does not hold, $\mathcal{J} \not\models^\rho C \sqsubseteq A$. If S is not surjective then this contradicts the fact that $\mathcal{I}^l$ is a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$.

Since $\mathcal{I}^{l+1}_\mathcal{R} \Rightarrow \mathcal{I}^l_\mathcal{R}$, we have that, for all concept names A, if $\rho_{\mathcal{I}_{l+1}} \in A^{\mathcal{I}_{l+1}}$, then $\rho_{\mathcal{I}_l} \in A^{\mathcal{I}_l}$. Now we show that any subsequence $\mathcal{I}^i, \ldots, \mathcal{I}^j$ of the sequence $\mathcal{I}^0, \ldots, \mathcal{I}^n$ such that $\rho_{\mathcal{I}^k} \in A^{\mathcal{I}^k}$ iff $\rho_{\mathcal{I}^{k+1}} \in A^{\mathcal{I}^{k+1}}$ holds for all concept names A and all $i \leq k < j$ has length bounded by $|C|(|\Sigma_\mathcal{T}| + |\mathcal{R}| + 1)$, for some $C \sqsubseteq A \in \mathcal{T}$. As $\rho_{\mathcal{I}^k} \in A^{\mathcal{I}^k}$ iff $\rho_{\mathcal{I}^{k+1}} \in A^{\mathcal{I}^{k+1}}$ and $\mathcal{I}^k$ is a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$, $i \leq k < j$, there is $C \sqsubseteq A \in \mathcal{T}$ such that for all k, $\mathcal{I}^k_\mathcal{R} \not\models^\rho C \sqsubseteq A$. By the argument of Lemma 30 the homomorphism $h : \mathcal{I}_C \to \mathcal{I}^k_\mathcal{R}$ is surjective. Then there are three possible cases: (1) $|\Delta^{\mathcal{I}^{k+1}}| > |\Delta^{\mathcal{I}^k}|$; (2) there is a role r in $\mathcal{I}^k$ which is replaced by s in $\mathcal{I}^{k+1}$ where $\mathcal{R} \models r \sqsubseteq s$ (and $\mathcal{R} \not\models s \sqsubseteq r$); or (3) there is $d \in \Delta^{\mathcal{I}^k} \setminus \{\rho_{\mathcal{I}^k}\}$ such that $d \in A^{\mathcal{I}^k}$ and $d \notin A^{\mathcal{I}^{k+1}}$. Since for all k, $h : \mathcal{I}_C \to \mathcal{I}^k_\mathcal{R}$ is surjective, Case 1 happens at most $|C|$. Also, as for all k, $|\Delta^{\mathcal{I}^k}| < |C|$ we have that Case 2 happens at most $|C| \cdot |\mathcal{R}|$ and Case 3 happens at most $|C| \cdot |\Sigma_\mathcal{T}|$. This gives a total bound of $|\mathcal{T}|(|\Sigma_\mathcal{T}| + |\mathcal{R}| + 1)$. ❑

Lemma 38 *For each counterexample received from the oracle in Line 6 of Algorithm 6, the number of recursive calls of the function "RefineCounterModel" in Algorithm 7 is bounded by $|\mathcal{T}|^2(|\Sigma_\mathcal{T}| + |\mathcal{R}| + 1)$.*

Proof. Let $\mathcal{I}^1, \mathcal{I}^2, \ldots, \mathcal{I}^n$ denote the sequence of $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$ in each recursive call of the function "RefineCounterModel". Lemma 36 guarantees that, for all $1 \leq k \leq n$, $|\Delta^{\mathcal{I}^k}|$ is bounded by $|\mathcal{T}|^2 \cdot (|\mathcal{R}| + 1)$. Let $\mathcal{I}'^k$ be the result of computing a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$ from $\mathcal{I}^k$, $1 \leq k \leq n$ (Line 2 of Algorithm 7). By construction of $\mathcal{I}'^k$, $\mathcal{I}'^k \Rightarrow (\mathcal{I}^k, d)$, for some $d \in \Delta^{\mathcal{I}^k}$. Now we make a case distinction:

1. d is not the root of $\mathcal{I}^k$: let $C \sqsubseteq A$ be a counterexample given by the oracle in Line 6 of Algorithm 6. Denote as $\mathcal{I}'^0$ the result of computing a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$ from $\mathcal{I}_C = \mathcal{I}^0$. By Lemma 30, $|\Delta^{\mathcal{I}'^0}| \leq |\mathcal{T}|$. So the depth of $\mathcal{I}'^0$ is bounded by $|\mathcal{T}|$. Notice that products do not increase the length of $\mathcal{EL}$ paths. This means that for all $1 \leq k \leq n$, depth of $\mathcal{I}^k$ is bounded by depth of $\mathcal{I}'^0$. Then, the number of times where d is not the root of $\mathcal{I}^k$ is bounded by $|\mathcal{T}|$.
2. d is the root of $\mathcal{I}^k$: let $\mathcal{I}^i, \mathcal{I}^{i+1}, \ldots, \mathcal{I}^j$ be a subsequence of $\mathcal{I}^1, \mathcal{I}^2, \ldots, \mathcal{I}^n$ such that for all $\mathcal{I}^l$, $i \leq l < j$, $\mathcal{I}'^l \Rightarrow \mathcal{I}^l$. That is, there is a simulation mapping the roots of $\mathcal{I}'^l$ and $\mathcal{I}^l$. By Case 1, there are at most $|\mathcal{T}|$ subsequences of this form. We first show that for all l, with $i \leq l < j$, we have that $\mathcal{I}^l_\mathcal{R} \not\Rightarrow \mathcal{I}^{l+1}_\mathcal{R}$ and $\mathcal{I}^{l+1}_\mathcal{R} \Rightarrow \mathcal{I}^l_\mathcal{R}$. As $\mathcal{I}^{l+1}$ is a single-edge tree shaped $\mathcal{I}'^l$-candidate for some $\mathcal{J}$ in $\mathfrak{I}$ (Line 11 of Algorithm 7), we have that $\mathcal{I}'^l_\mathcal{R} \not\Rightarrow \mathcal{I}^{l+1}_\mathcal{R}$ and $\mathcal{I}^{l+1}_\mathcal{R} \Rightarrow \mathcal{I}'^l_\mathcal{R}$, where $\mathcal{I}'^l$ is the $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$ of $\mathcal{I}^l$ (computed in Line 2 of Algorithm 7). The assumption in this case states that for all $\mathcal{I}^l$, $i \leq l < j$, $\mathcal{I}'^l \Rightarrow \mathcal{I}^l$. By construction of $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodels for $\mathcal{H}$, $\mathcal{I}'^l$ is a subinterpretation of $\mathcal{I}^l$. Then, we have that $\mathcal{I}^l_\mathcal{R} \not\Rightarrow \mathcal{I}^{l+1}_\mathcal{R}$ and $\mathcal{I}^{l+1}_\mathcal{R} \Rightarrow \mathcal{I}^l_\mathcal{R}$. By Lemma 37, the length of a subsequence is bounded by $|\mathcal{T}|(|\Sigma_\mathcal{T}| + |\mathcal{R}| + 1)$, which gives the total bound of $|\mathcal{T}|^2(|\Sigma_\mathcal{T}| + |\mathcal{R}| + 1)$ recursive calls.

❑

Now, termination in polynomial time is a consequence of Lemma 39 below.

Lemma 39 *Let $\mathfrak{I}$ be a sequence 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 at most $|\mathcal{T}|(|\Sigma_\mathcal{T}| + |\mathcal{R}| + 1)$ often with a new interpretation.

The rest of the section is devoted to proving Lemma 39. For easy reference, assume that at each point of the execution of the algorithm, $\mathfrak{J}$ has the form $\mathcal{J}^0, \ldots, \mathcal{J}^k$ for some $k \geq 0$. To establish Point (i) above, we closely follow (Angluin, Frazier, and Pitt 1992) and show that

(iii) for every $\mathcal{J}^i$, there is a $D_i \sqsubseteq A_i \in \mathcal{T}$ with $\mathcal{J}^i_{\mathcal{R}} \not\models^\rho D_i \sqsubseteq A_i$ 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 $\mathcal{J}^i$ is added to $\mathfrak{J}$ in the algorithm, then $\mathcal{J}^i$ is a $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$. For a multi-edge tree shaped interpretation $\mathcal{I}_\mathcal{R}$ and a concept inclusion $C \sqsubseteq A$, we write $\mathcal{I}_\mathcal{R} \models^\rho C \sqsubseteq A$ if $\rho_{\mathcal{I}_\mathcal{R}} \notin C^{\mathcal{I}_\mathcal{R}}$ or $\rho_{\mathcal{I}_\mathcal{R}} \in A^{\mathcal{I}_\mathcal{R}}$; that is, the inclusion $C \sqsubseteq A$ is satisfied at the root of $\mathcal{I}_\mathcal{R}$, but not necessarily at other points in $\mathcal{I}_\mathcal{R}$. It is easy to see that if some interpretation $\mathcal{I}$ is a $(\mathcal{T}, \mathcal{R})$-countermodel, then there is a $C \sqsubseteq A \in \mathcal{T}$ such that $\mathcal{I}_\mathcal{R} \not\models^\rho C \sqsubseteq A$. To prove Point (iv), we need to show that at no time there are $\mathcal{J}^j, \mathcal{J}^k \in \mathfrak{J}$ refuting the same CI in $\mathcal{T}$. Suppose this is not the case and both $\mathcal{J}^j, \mathcal{J}^k \in \mathfrak{J}$ refute $C \sqsubseteq A \in \mathcal{T}$. Without loss of generality assume $j < k$. By Lemma 41, if $\mathcal{J}^k_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $j < k$, then $\rho_{\mathcal{J}^j_\mathcal{R}} \notin C^{\mathcal{J}^j_\mathcal{R}}$. This contradicts the assumption that both $\mathcal{J}^j, \mathcal{J}^k \in \mathfrak{J}$ refute $C \sqsubseteq A \in \mathcal{T}$ in their roots. Thus, the length of $\mathfrak{J}$ is bounded by the number of CIs in $\mathcal{T}$. To prove Lemma 41, we first establish the intermediate Lemma 40 below.

Lemma 40 *If the interpretation $\mathcal{I}$ constructed in Line 2 of Algorithm 7 satisfies $\mathcal{I}_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $\rho_{\mathcal{J}^j_\mathcal{R}} \in C^{\mathcal{J}^j_\mathcal{R}}$ for some j, then $\mathcal{J} = \mathcal{J}^k$ is replaced with $\mathcal{J}'$ in Line 11 of Algorithm 6 for some $k \leq j$ or $\mathcal{I}$ is refined in Line 11 of Algorithm 7 to some $\mathcal{I}'$ such that $\mathcal{T} \models C_{\mathcal{I}'} \sqsubseteq A$ and $\mathcal{H} \not\models C_{\mathcal{I}'} \sqsubseteq A$.*

Proof. Assume that the interpretation $\mathcal{I}$ constructed in Line 2 of Algorithm 7 satisfies $\mathcal{I}_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and that there is some $\mathcal{J}^j \in \mathfrak{J}$ such that $\rho_{\mathcal{J}^j} \in C^{\mathcal{J}^j}$. If there is some $k < j$ such that the algorithm replaces $\mathcal{J}^k$ then we are done. Thus assume that there is no such k. We aim to show that $\mathcal{J} = \mathcal{J}^j$ is replaced with $\mathcal{J}'$ in Line 11 of Algorithm 6 or $\mathcal{I}$ is refined in Line 11 of Algorithm 7 to $\mathcal{I}'$. To this end, it suffices to prove that (a) $\mathcal{J}^j_\mathcal{R} \not\Rightarrow (\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R})$ and (b) either :

- $(\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R})_\mathcal{R} \not\models \mathcal{T}$ and $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R}$ is single-edge tree shaped, where $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R}$ is the result of multi-edge reducing $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R}$, or;
- $\mathcal{T} \models C_{\mathcal{I}^\times} \sqsubseteq A$ and $\mathcal{H} \not\models C_{\mathcal{I}^\times} \sqsubseteq A$, where $\mathcal{I}^\times$ is a single-edge tree shaped $\mathcal{I}$-candidate for $\mathcal{J}$.

To show (a) assume to the contrary that $\mathcal{J}^j_\mathcal{R} \Rightarrow (\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R})$. We establish a contradiction against $\mathcal{I}_\mathcal{R} \models \mathcal{H}$ (which holds by construction of $\mathcal{I}$ in the algorithm) by showing that

1. $\mathcal{I}_\mathcal{R} \not\models^\rho C_{\mathcal{J}^j} \sqsubseteq A$ and
2. $C_{\mathcal{J}^j} \sqsubseteq A \in \mathcal{H}$.

For Point 1, $\mathcal{J}^j_\mathcal{R} \Rightarrow (\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R})$ and $\rho_{\mathcal{J}^j} \in (C_{\mathcal{J}^j})^{\mathcal{J}^j}$ imply $\rho_{\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R}} \in (C_{\mathcal{J}^j})^{\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R}}$, which gives $\rho_{\mathcal{I}_\mathcal{R}} \in (C_{\mathcal{J}^j})^{\mathcal{I}_\mathcal{R}}$, by Lemma 20 and Lemma 18. It remains to observe that $\mathcal{I}_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ implies $\rho_{\mathcal{I}_\mathcal{R}} \notin A^{\mathcal{I}_\mathcal{R}}$. In view of the construction of $\mathcal{H}$ in the algorithm, Point 2 can be established by showing that $\mathcal{T} \models C_{\mathcal{J}^j} \sqsubseteq A$. Since $C \sqsubseteq A \in \mathcal{T}$, it suffices to prove that $\mathcal{R} \models C_{\mathcal{J}^j} \sqsubseteq C$. This, however, is an immediate consequence of the assumption in this lemma which says that $\rho_{\mathcal{J}^j_\mathcal{R}} \in C^{\mathcal{J}^j_\mathcal{R}}$.

Now we want to show (b). The fact that $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R} \not\models \mathcal{T}$ is a consequence of $\mathcal{I}_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $\rho_{\mathcal{J}^j_\mathcal{R}} \in C^{\mathcal{J}^j_\mathcal{R}}$. We know that $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R}$ is the result of removing redundancies from $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R}$. So $(\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R})_\mathcal{R} \not\models \mathcal{T}$ iff $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R} \not\models \mathcal{T}$. If $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R}$ is single-edge tree shaped then Algorithm 6 replaces $\mathcal{J}^j$ with $\mathcal{J}'$ in Line 11. Otherwise we need to show that $\mathcal{T} \models C_{\mathcal{I}^\times} \sqsubseteq A$ and $\mathcal{H} \not\models C_{\mathcal{I}^\times} \sqsubseteq A$, where $\mathcal{I}^\times$ is the result of restrict saturating and unfolding $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^j_\mathcal{R}$. Since $\mathcal{I}_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $\rho_{\mathcal{J}^j_\mathcal{R}} \in C^{\mathcal{J}^j_\mathcal{R}}$ we know that $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$. By definition of $\mathcal{I}^\times$ we have that $\mathcal{I}_\mathcal{R} \times_\rho \mathcal{J}^j_\mathcal{R} \Rightarrow \mathcal{I}^\times_\mathcal{R}$. Then, $\rho_{\mathcal{I}^\times_\mathcal{R}} \in C^{\mathcal{I}^\times_\mathcal{R}}$. As $\mathcal{I}^\times_\mathcal{R} \Rightarrow \mathcal{I}_\mathcal{R}$ and $\rho_{\mathcal{I}_\mathcal{R}} \notin A^{\mathcal{I}_\mathcal{R}}$, we have that $\rho_{\mathcal{I}^\times_\mathcal{R}} \notin A^{\mathcal{I}^\times_\mathcal{R}}$. Then, $\mathcal{I}^\times_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$. That is, $\mathcal{T} \models C_{\mathcal{I}^\times} \sqsubseteq A$. As $\mathcal{I}$ is a $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$, $\mathcal{H} \not\models C_\mathcal{I} \sqsubseteq A$. Since $\mathcal{I}^\times_\mathcal{R} \Rightarrow \mathcal{I}_\mathcal{R}$, we have that $\mathcal{H} \not\models C_{\mathcal{I}^\times} \sqsubseteq A$. ❑

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

Lemma 41 *At any time of the algorithm execution, the following condition holds: if $\mathcal{J}^k_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $j < k$, then $\rho_{\mathcal{J}^j_\mathcal{R}} \notin C^{\mathcal{J}^j_\mathcal{R}}$.*

Proof. We prove the invariant formulated in Lemma 41 by induction on the number of iterations of the while loop in Algorithm 6. Clearly, the invariant is satisfied before the loop is entered. We now consider the two places where $\mathfrak{J}$ is modified, that is, Line 11 and Line 13 of Algorithm 6, starting with the latter. Let $\mathcal{I}$ be a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$.

In Line 13, $\mathcal{I}$ is appended to $\mathfrak{J}$. Assume that $\mathcal{I}_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$. We have to show that, before $\mathcal{I}$ was added to $\mathfrak{J}$, there was no $\mathcal{J}^j \in \mathfrak{J}$ with $\rho_{\mathcal{J}^j_\mathcal{R}} \in C^{\mathcal{J}^j_\mathcal{R}}$. This, however, is immediate by Lemmas 40 and 38, which shows that the number of recursive calls is polynomial and, therefore, the sequence is updated in a polynomial number of steps if the conditions for Lemma 40 are satisfied.

Now suppose that the algorithm replaces some $\mathcal{J}^k \in \mathfrak{J}$ with $\mathcal{J}^{k\prime}$ in Line 11. Suppose by way of contradiction that the lemma does not hold. Then, either:

1. there is $j < k$ such that $\mathcal{J}^{k\prime}{}_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$ and $\rho_{\mathcal{J}^j_\mathcal{R}} \in C^{\mathcal{J}^j_\mathcal{R}}$ or;
2. there is some $j > k$ such that $\rho_{\mathcal{J}^{k\prime}{}_\mathcal{R}} \in C^{\mathcal{J}^{k\prime}{}_\mathcal{R}}$ and $\mathcal{J}^j_\mathcal{R} \not\models^\rho C \sqsubseteq A \in \mathcal{T}$.

Suppose Case 1 holds. Let $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^k_\mathcal{R}$ be the maximal multi-edge reduced tree shaped interpretation that is contained in $\mathcal{I}_\mathcal{R} \times \mathcal{J}^k_\mathcal{R}$ with root $(\rho_\mathcal{I}, \rho_{\mathcal{J}^k})$. Since $\mathcal{J}^{k\prime}$ is obtained from $\mathcal{I}_\mathcal{R} \times^{\mathsf{r}}_\rho \mathcal{J}^k_\mathcal{R}$ by removing subtrees (see Lemma 31),

${\mathcal{J}^{k}}'_{\mathcal{R}} \not\models^{\rho} C \sqsubseteq A$ implies $(\mathcal{I}_{\mathcal{R}} \times^{\mathsf{r}}_{\rho} \mathcal{J}^{k}_{\mathcal{R}})_{\mathcal{R}} \not\models^{\rho} C \sqsubseteq A$. Consequently, $\mathcal{I}_{\mathcal{R}} \not\models^{\rho} C \sqsubseteq A$ or $\mathcal{J}^{k}_{\mathcal{R}} \not\models^{\rho} C \sqsubseteq A$. The former and Lemma 40 implies that there is $l \leq j < k$ such that $\mathcal{J}^{l}$ was refined instead of $\mathcal{J}^{k}$, a contradiction. If $\mathcal{J}^{k}_{\mathcal{R}} \not\models^{\rho} C \sqsubseteq A$ then, since $\rho_{\mathcal{J}^{j}_{\mathcal{R}}} \in C^{\mathcal{J}^{j}_{\mathcal{R}}}$, this contradicts the assumption that Lemma was true before $\mathcal{J}^{k}$ was replaced by ${\mathcal{J}^{k}}'$.

Now, suppose Case 2 holds. If $\rho_{{\mathcal{J}^{k}}'_{\mathcal{R}}} \in C^{{\mathcal{J}^{k}}'_{\mathcal{R}}}$ then $\rho_{\mathcal{J}^{k}_{\mathcal{R}}} \in C^{\mathcal{J}^{k}_{\mathcal{R}}}$. Since $\mathcal{J}^{j}_{\mathcal{R}} \not\models^{\rho} C \sqsubseteq A \in \mathcal{T}$ and $j > k$, again this contradicts the assumption that the Lemma was true before $\mathcal{J}^{k}$ was replaced by ${\mathcal{J}^{k}}'$. Thus in either case Lemma 41 holds. ❑

We now turn towards proving Point (ii) above. It is a consequence of Lemma 42 below.

Lemma 42 *Let $\mathfrak{I}$ be a sequence computed at some point of an execution of Algorithm 6. Then each $\mathcal{J} \in \mathfrak{I}$ is replaced at most $|\mathcal{T}|(|\Sigma_{\mathcal{T}}| + |\mathcal{R}| + 1)$ often with a new interpretation.*

Proof. Let $\mathcal{J}^0, \ldots, \mathcal{J}^n$ be the sequence of interpretations such that $\mathcal{J}^{l+1}$ replaces $\mathcal{J}^{l}$ in Line 11 of Algorithm 6, $l < n$. We first show that $\mathcal{J}^{l}_{\mathcal{R}} \not\Rightarrow \mathcal{J}^{l+1}_{\mathcal{R}}$ and $\mathcal{J}^{l+1}_{\mathcal{R}} \Rightarrow \mathcal{J}^{l}_{\mathcal{R}}$. Since $\mathcal{J}^{l+1}$ is a subinterpretation of some $\mathcal{I}_{\mathcal{R}} \times_{\rho} \mathcal{J}^{l}_{\mathcal{R}}$ and $\mathcal{I}_{\mathcal{R}} \times_{\rho} \mathcal{J}^{l}_{\mathcal{R}} \Rightarrow \mathcal{J}^{l}_{\mathcal{R}}$ we obtain that $\mathcal{J}^{l+1}_{\mathcal{R}} \Rightarrow \mathcal{J}^{l}_{\mathcal{R}}$. Also, by Line 7 of Algorithm 7, $\mathcal{J}^{l}_{\mathcal{R}} \not\Rightarrow \mathcal{I}_{\mathcal{R}} \times_{\rho} \mathcal{J}^{l}_{\mathcal{R}}$, then, $\mathcal{J}^{l}_{\mathcal{R}} \not\Rightarrow \mathcal{J}^{l+1}_{\mathcal{R}}$, where $\mathcal{J}^{l+1}_{\mathcal{R}}$ is the subinterpretation of $\mathcal{I}_{\mathcal{R}} \times_{\rho} \mathcal{J}^{l}_{\mathcal{R}}$ computed by Line 10 of Algorithm 6. By Lines 10 and 2 of Algorithm 6, for all $l < n$, $\mathcal{J}^{l}$ is a $\mathcal{T}$-essential $(\mathcal{T}, \mathcal{R})$-countermodel for $\mathcal{H}$. Then we can apply Lemma 37 to obtain that each $\mathcal{J} \in \mathfrak{I}$ is replaced at most $|\mathcal{T}|(|\Sigma_{\mathcal{T}}|+|\mathcal{R}|+1)$ often with a new interpretation. ❑

Theorem 43 *The learning framework $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{lhs}})$ is polynomial time exact learnable.*

Proofs for Theorem 2

For convenience of the reader we state again our definition of positive polynomial query reducibility and also Theorem 2.

Definition 44 *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 $f_{MEM} : X' \to X$ and a partial function $f_{EQ} : \mathcal{L} \times \mathcal{L} \times X \to X'$, defined for every (l, h, x) such that $|h| \leq p_1(|l|)$, for which the following conditions hold:*

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

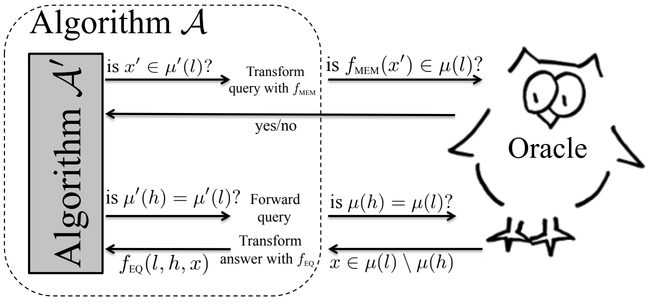


Figure 6: Learning by reduction algorithm

Theorem 2 (restated). 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.

Proof. Let A' be a polynomial learning algorithm for $(X', \mathcal{L}, \mu')$, which only uses positive bounded equivalence queries. We construct a learning algorithm A for $(X, \mathcal{L}, \mu)$, using internally the learning algorithm A', as follows. As learning $(X, \mathcal{L}, \mu)$ positively polynomially query reduces to learning $(X', \mathcal{L}, \mu')$, we have that:

- for any $l, h \in \mathcal{L}$, $\mu(h) \subseteq \mu(l)$ if, and only if, $\mu'(h) \subseteq \mu'(l)$; and
- there are functions $f_{\mathsf{MEM}} : X' \to X$ and $f_{\mathsf{EQ}} : \mathcal{L} \times \mathcal{L} \times X \to X'$ such that f_{MEM} transforms a membership query with '$x' \in X'$' as input into a query '$x \in X$' and f_{EQ} transforms a positive counterexample '$x \in X$' into a positive counterexample '$x' \in X'$'.

So, whenever a membership query with $x' \in X'$ as input is called by A' we compute $f_{\mathsf{MEM}}(x')$ and call the $\mathsf{MEM}_{l,X}$ oracle. We return 'yes' to A' if $f_{\mathsf{MEM}}(x') \in \mu(l)$ and 'no' otherwise, see Fig. 6. Whenever an equivalence query with $h \in \mathcal{L}$ as input is called by A' we pass it on to the $\mathsf{EQ}_{l,X}$ oracle. If it returns 'yes' then the learner succeeded. Otherwise the oracle returns 'no' and provides a counterexample $x \in X$. Notice that, since A' is a positive learning algorithm, $\mu'(h) \subseteq \mu'(l)$ and then $\mu(h) \subseteq \mu(l)$ by definition of a positive polynomial reduction. Thus, any counterexample x can only be positive, that is, $x \in \mu(l) \setminus \mu(h)$. Then, we compute $x' = g(l, h, x)$ and return it to A'. Notice that computing $f_{\mathsf{EQ}}(l, h, x)$ may require posing additional membership queries (recall that l can only be accessed via queries to the oracle $\mathsf{MEM}_{l,X}$), not shown on Fig. 6.

By definition of f_{MEM} and f_{EQ} all the answers provided to A' are consistent with answers the oracles $\mathsf{MEM}_{l,X'}$ and $\mathsf{EQ}_{l,X'}$ would provide to A'. Clearly, if algorithm A terminates then it learns l.

It remains to prove the polynomial query bound for A. Let $p_1(\cdot)$, $p_2(\cdot)$ and $p_3(\cdot, \cdot)$ be the polynomials of Definition 44, that is,

- $p_1(|l|)$ is the polynomial bound on $|h|$;
- $|f_{\mathsf{MEM}}(x')| \leq p_2(|x'|)$;

- $p_3(|l|,|x|)$ is the polynomial query bound for computing $f_{\mathsf{EQ}}(l,h,x)$.

Let $p(\cdot,\cdot)$ be such a polynomial that in every run of A', the sum of the sizes of inputs to queries used by A' up to each step of computation is bounded by $p(|l|,|y'|)$, where $|l|$ is the size of the target $l \in \mathcal{L}$ and $|y'|$ is the size of the largest counterexample $y' \in X'$ seen by A' up to that point of computation. As y' is the result of transformation with function f_{EQ} of some counterexample $y \in X$ given by the $\mathsf{EQ}_{l,X}$ oracle to algorithm A, its size $|y'|$ is bounded by $p_3(|l|,|y|)$. Notice that y is also the largest counterexample seen so far by A. Thus, at every step of computation the sum of the sizes of inputs to queries used to run A' up to that step is bounded by a polynomial $p'(|l|,|y|) = p(|l|,p_3(|l|,|y|))$.

For every membership query with $x' \in X'$ asked by A', the size of x' does not exceed the polynomial query bound of A' up to that point, that is, $|x'| \leq p'(|l|,|y|)$. Then, the sum of the sizes of inputs to queries needed to transform answers to equivalence queries is bounded by $p_2(p'(|l|,|y|))$ and $p_3(|l|,|y|)$, respectively. All in all, at every step of computation the sum of the sizes of inputs to queries used by A up to that step is bounded by $p'(|l|,|y|) \cdot p_2(p'(|l|,|y|)) \cdot p_3(|l|,|y|)$ steps, which is polynomial in $|l|$ and $|y|$, as required. ❑

We now modify the notion of a positive polynomial *query* reduction to obtain an appropriate notion of positive polynomial *time* reduction that enables us to prove polynomial time exact learnability results in the data retrieval framework by reduction to the subsumption framework.

Definition 45 *A learning framework $\mathfrak{F} = (X,\mathcal{L},\mu)$ positively polynomial time 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 $f_{\mathsf{MEM}} : X' \to X$ and a partial function $f_{\mathsf{EQ}} : \mathcal{L} \times \mathcal{L} \times X \to X'$, defined for every (l,h,x) such that $|h| \leq p_1(|l|)$, for which the following conditions hold:*

- *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 $f_{\mathsf{EQ}}(l,h,x) \in \mu'(l) \setminus \mu'(h)$;*
- *$f_{\mathsf{MEM}}(x')$ is computable in time $p_2(|x'|)$;*
- *$f_{\mathsf{EQ}}(l,h,x)$ is computable in time $p_3(|l|,|x|)$ and l can only be accessed by calls to the membership oracle $\mathsf{MEM}_{l,X}$.*

Note that for time reductions the conditions for query reductions that $|f_{\mathsf{MEM}}(x')| \leq p_2(|x'|)$ and $|f_{\mathsf{EQ}}(l,h,x)| \leq p_3(|l|,|x|)$ are a consequence of the polynomial time bound for computing $f_{\mathsf{MEM}}(x')$ and $f_{\mathsf{EQ}}(l,h,x)$. The proof of Theorem 2 above can be easily modified to show the following.

Theorem 46 *Let $\mathfrak{F} = (X,\mathcal{L},\mu)$ and $\mathfrak{F}' = (X',\mathcal{L},\mu')$ be proper learning frameworks. If there exists a polynomial time reduction from $\mathfrak{F}$ to $\mathfrak{F}'$ and $\mathfrak{F}'$ is polynomial time exact learnable then $\mathfrak{F}$ is polynomial time exact learnable.*

Proofs for Polynomial Query Reduction of $\mathfrak{F}_{\mathcal{D}}$(DL-Lite$^{\exists}_{\mathcal{H}}$, $\mathcal{ELI}$-IQ)

In this section we prove that $\mathfrak{F}_{\mathcal{D}}$(DL-Lite$^{\exists}_{\mathcal{H}}$, $\mathcal{ELI}$-IQ) is polynomial query learnable. Before we start our proofs, we first recall the notions of role saturation and parent/child merging used in Line 2 of Algorithm 1.

(Role saturation) if $(\mathcal{T},\mathcal{A}) \models C'(a)$ and C' is the result of replacing a role r in the labeled tree of C by some role r', where $\mathcal{T} \models r' \sqsubseteq r$ and $r' \not\equiv_{\mathcal{T}} r$, then consider $(\mathcal{A}, C'(a))$ instead of $(\mathcal{A}, C(a))$ as a counterexample.

(Parent/child merging) if, for nodes n_1,n_2,n_3 in the labeled tree T_C of C where n_2 is an r-successor of n_1, n_3 is an s-successor of n_2 and $r^- \equiv_{\mathcal{T}} s$, we have that $(\mathcal{T},\mathcal{A}) \models C'(a)$ with C' as the result of identifying n_1 and n_3; then consider $(\mathcal{A}, C'(a))$ instead of $(\mathcal{A}, C(a))$ as counterexample.

Lemma 3 (restated). 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 and 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 $(\mathcal{T},\mathcal{A}') \models C(a)$.

Proof. For Point (1), assume that C is a basic concept B. By Lemma 13, if $(\mathcal{T},\mathcal{A}) \models B(a)$ then $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models B(a)$, where $\mathcal{I}_{\mathcal{T},\mathcal{A}} = \bigcup \mathcal{I}_{n \geq 0}$ is the canonical model of $(\mathcal{T},\mathcal{A})$. Point (1) of this lemma follows from Claim 1 below.

Claim 1. For all n, if $a \in B^{\mathcal{I}_n}$, where B is a basic concept, then there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $a \in B^{\mathcal{I}_{\mathcal{T},\mathcal{A}'}}$.

For $n = 0$, as $\mathcal{I}_0 = \mathcal{I}_{\mathcal{A}}$, if $\mathcal{I}_0 \models B(a)$ then we are done. Otherwise, there is $k \leq n$ such that $\mathcal{I}_k \not\models B(a)$ and $\mathcal{I}_{k+1} \models B(a)$. By construction of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, there is $B' \sqsubseteq D \in \mathcal{T}$ such that $a \in (B' \setminus D)^{\mathcal{I}_k}$, $a \in D^{\mathcal{I}_{k+1}}$ and $\mathcal{R} \models D \sqsubseteq B$ (where $\mathcal{R}$ is the set of RIs in $\mathcal{T}$). As $\mathcal{T}$ is a DL-Lite$^{\exists}_{\mathcal{H}}$ TBox, B' is a basic concept. Suppose the claim holds for $a \in B'^{\mathcal{I}_k}$. Then, there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T},\mathcal{A}') \models B'(a)$. As $B' \sqsubseteq D \in \mathcal{T}$ and $\mathcal{R} \models D \sqsubseteq B$, $(\mathcal{T},\mathcal{A}') \models B(a)$, as required.

We now prove Point (2). Assume C is of the form $\exists r.C'$ with r a role. If $(\mathcal{T},\mathcal{A}) \models \exists r.C'(a)$ then, by Lemma 13, $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models \exists r.C'(a)$. By semantics of $\exists$, there is $d \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ such that $(a,d) \in r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and $d \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. If $d \in \Delta^{\mathcal{I}_{\mathcal{A}}}$ then there is $s(a,d) \in \mathcal{A}$ such that $\mathcal{T} \models s \sqsubseteq r$ and $(\mathcal{T},\mathcal{A}) \models C'(d)$ and, so, we are done. Otherwise, we have that $d \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{A}}}$. By Lemma 11, there is a homomorphism $h : \mathcal{I}_{C'} \to \mathcal{I}_{\mathcal{T},\mathcal{A}}$ mapping $\rho_{C'}$ to d. To simplify notation, consider labeled tree $T_{C'}$ (isomorphic to $\mathcal{I}_{C'}$) as the domain of h. Let $\mathsf{Im}_h = \{e \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \mid h(n) = e \text{ for some } n \text{ in } T_{C'}\}$ be the image of h.

Claim 2. If $\exists r.C'(a)$ is role saturated and parent/child merged then $\mathsf{Im}_h \subseteq \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{A}}}$.

We can assume that C' is not $\top$ (otherwise $\exists r.C'$ is a basic concept, which is treated in Point (1)). Then, the root

$\rho_{T_{C'}}$ of $T_{C'}$ has an s-successor n which is either mapped to a or to a child of d in $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{A}}}$. If $h(n) = a$ then, as $(a, d) \in r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and $\exists r.C'(a)$ is role saturated, we have that $r \equiv_{\mathcal{T}} s^-$. Then $(\mathcal{T}, \mathcal{A}) \models C''(a)$, where C'' is the result of identifying n with the root of the tree corresponding to $\exists r.C'$. This contradicts the fact that $\exists r.C'(a)$ is parent/child merged. Otherwise, n is mapped to a child of d in $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{A}}}$. In this case, suppose to the contrary that there is n' in $T_{C'}$ such that $h(n') \in \Delta^{\mathcal{I}_{\mathcal{A}}}$. This can only be if there are 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 $h(n_1) = h(n_3)$. By construction of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, there is a role t such that $(h(n_1), h(n_2)) \in t^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and for all roles t' if $(h(n_1), h(n_2)) \in t'^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ then $\mathcal{T} \models t \sqsubseteq t'$. By role saturation, $r' \equiv_{\mathcal{T}} t$ and $s'^- \equiv_{\mathcal{T}} t$, which means that $r' \equiv_{\mathcal{T}} s'^-$. Since n_1, n_3 are distinct nodes in $T_{C'}$, this contradicts the fact that $\exists r.C'(a)$ is parent/child merged.

By Claim 2 the whole image of h is outside of $\Delta^{\mathcal{I}_{\mathcal{A}}}$. Then, by construction of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, there is $B \sqsubseteq D \in \mathcal{T}$ such that $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models B(a)$ and $\mathcal{T} \models D \sqsubseteq \exists r.C'$. By Claim 1 above, if $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models B(a)$ then there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}') \models B(a)$. Since $B \sqsubseteq D \in \mathcal{T}$ and $\mathcal{T} \models D \sqsubseteq \exists r.C'$ we have that $(\mathcal{T}, \mathcal{A}') \models \exists r.C'(a)$, as required. ❏

Lemma 4 (restated). 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. Let $(\mathcal{A}, C(a))$ be the counterexample given to the function "ReduceCounterExample" (Line 1). In Line 2 of Algorithm 1, we exhaustively apply the rules role saturation and parent/child merging. Let $(\mathcal{A}, C^*(a))$ denote the counterexample obtained by the application of these rules. If C^* is of the form $C_0^* \sqcap ... \sqcap C_n^*$ then there is a C_i^*, $0 \leq i \leq n$, such that $(\mathcal{A}, C_i^*(a))$ is also a counterexample. In Line 4, Algorithm 1, chooses a conjunct C_i^* such that $(\mathcal{A}, C_i^*(a))$ is a counterexample. Otherwise, C^* is a basic concept or C^* is of the form $\exists r.C'$. In this case consider $C_i^* = C^*$. Now, we make a case distinction:

- C_i^* is a basic concept B: then Point (1) of Lemma 3 applies.
- C_i^* is of the form $\exists r.C'$ and we make a case distinction:
  1. there is $r(a, b) \in \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}) \models C'(b)$. In this case, Algorithm 1 recursively calls the function "ReduceCounterExample" (Line 8).
  2. otherwise, notice that since $C^*(a)$ is role saturated and parent/child merged, $C_i^*(a)$ also have the same properties. Then, Point (2) of Lemma 3 applies.

In case (1), with $(\mathcal{A}, C'(b))$ as the counterexample, observe that $(\mathcal{A}, C'(b))$ is a refined version of the initial counterexample $(\mathcal{A}, C(a))$. Since the size of the refined counterexample is strictly smaller in every recursive call of the function "ReduceCounterExample", the number of calls to this function is bounded by $|C|$. Then, conditions for Point (1) or Point (2) of Lemma 3 are satisfied after polynomially many recursive calls.

It remains to show that each call of the function "ReduceCounterExample" is computable with polynomial query bound with respect to $|\mathcal{T}|$, $|C|$ and $|\mathcal{A}|$. Note that by assumption $|\mathcal{H}|$ is polynomial in $|\mathcal{T}|$. Algorithm 1 makes membership queries in Lines 2, 7 and 11. In Line 2 the sum of inputs of membership queries is bounded by $|C|^3 \cdot |\Sigma_{\mathcal{T}}|$. In Line 7 it is verified for every $s(a, b) \in \mathcal{A}$ such that $\mathcal{T} \models s \sqsubseteq r$ whether $(\mathcal{T}, \mathcal{A}) \models C'(b)$. So the sum of inputs of membership queries is bounded by $|\mathcal{A}| \cdot |C|$. Finally, in Line 11 the sum of inputs of membership queries is bounded by $|\mathcal{A}_a| \cdot |C|$, where $\mathcal{A}_a \subseteq \mathcal{A}$ is the set of assertions in $\mathcal{A}$ where individual a occurs. ❏

We now detail the polynomials of Definition 44 involved in our proof for DL-Lite$^{\exists}_{\mathcal{H}}$. Let $\mathcal{T}$ be the target DL-Lite$^{\exists}_{\mathcal{H}}$ TBox with signature $\Sigma_{\mathcal{T}}$. We want to define $p_1(\cdot), p_2(\cdot,\cdot), p_3(\cdot,\cdot)$ such that:

- $p_1(|\mathcal{T}|)$ is the polynomial bound on $|\mathcal{H}|$;
- $p_2(|A \sqsubseteq C|)$ is the polynomial bound on $|f_{\mathsf{MEM}}(A \sqsubseteq C)|$;
- $p_3(|\mathcal{T}|, |(\mathcal{A}, D(a))|)$ is the polynomial query bound for computing $f_{\mathsf{EQ}}(\mathcal{T}, \mathcal{H}, (\mathcal{A}, D(a)))$.

In the learning algorithm for DL-Lite$^{\exists}_{\mathcal{H}}$ presented in (Konev et al. 2014), $\mathcal{H}$ is the union of $\mathcal{H}_{add}$ and $\mathcal{H}_{basic}$. We have that $|\mathcal{H}_{basic}|$ is bounded by $2 \cdot |\Sigma_{\mathcal{T}}|^2$ (combinations of two in the signature plus inverse roles) and $\mathcal{H}_{add}$ has at most $|\Sigma_{\mathcal{T}}|$ CIs. The left hand side of each CI in $\mathcal{H}_{add}$ has size 1 and the right hand side of each CI is bounded by $|\Sigma_{\mathcal{T}}| \cdot |\mathcal{T}|$. Then, $|\mathcal{H}_{add}| \leq |\Sigma_{\mathcal{T}}|(|\Sigma_{\mathcal{T}}| \cdot |\mathcal{T}| + 1)$. As $|\Sigma_{\mathcal{T}}| \leq |\mathcal{T}|$, we have $|\mathcal{H}| \leq p_1(|\mathcal{T}|)$ with $p_1(|\mathcal{T}|) = |\mathcal{T}|^3 + 2 \cdot |\mathcal{T}|^2 + |\mathcal{T}|$ (we note that normally the size of the signature is much smaller than the size of the TBox and, so, this rough estimate only serves for the purpose of defining our polynomials). As $f_{\mathsf{MEM}}(A \sqsubseteq C) = (\{A(a)\}, C(a))$, we have $p_2(|A \sqsubseteq C|) = |C| + 1$. By the proof of the lemma above, we have that $p_3(|\mathcal{T}|, |(\mathcal{A}, D(a))|) =$

$$k \cdot \underbrace{|D|}_{\text{recursive calls}} \;\big(\; \overbrace{(|D|^3 \cdot |\mathcal{T}|)}^{\text{role sat., parent/child}} + \overbrace{|\mathcal{A}| \cdot |D|}^{\text{queries on } \mathcal{T}} \big)$$

for some constant $k \in \mathbb{N}$.

Proofs for Polynomial Time Reduction of $\mathfrak{F}_{\mathcal{D}}(\mathcal{ELH}_{\mathsf{lhs}}, \mathbf{AQ})$

We now prove polynomial time learnability for the learning framework $\mathfrak{F}_{\mathcal{D}}(\mathcal{ELH}_{\mathsf{lhs}}, \text{AQ})$. We employ Theorem 46 in our polynomial time reduction to $\mathfrak{F}_{\mathcal{S}}(\mathcal{ELH}_{\mathsf{lhs}})$. Before we start our proofs, we first define the notion of a *minimal* ABox $\mathcal{A}$ and show in Lemma 47 that the size of $\mathcal{A}$ is polynomially bounded by $|\mathcal{T}|$. Recall that we denote by $\mathcal{A}^{-a}$ the result of removing from $\mathcal{A}$ all ABox assertions where a occurs. That is, $\mathcal{A}^{-a} = \mathcal{A} \setminus \mathcal{A}^a$, where $\mathcal{A}^a = \{r(a, b) \mid b \in \mathsf{Ind}(\mathcal{A}), r \in \Sigma_{\mathcal{T}} \cap \mathsf{N_R}\} \cup \{r(b, a) \mid b \in \mathsf{Ind}(\mathcal{A}), r \in \Sigma_{\mathcal{T}} \cap \mathsf{N_R}\} \cup \{A(a) \mid A \in \Sigma_{\mathcal{T}} \cap \mathsf{N_C}\}$. Also, $\mathcal{A}^{-r(a,b)}$ is obtained by removing a role assertion $r(a, b)$ from $\mathcal{A}$. Let $\mathcal{I}_{\mathcal{A}}$ be the canonical model of an ABox $\mathcal{A}$. We say that $\mathcal{I}_{\mathcal{A}}$ is a *countermodel* if $\mathcal{I}_{\mathcal{A}} \not\models \mathcal{T}$ and $\mathcal{I}_{\mathcal{A}} \models \mathcal{H}$. We define $\mathcal{A}$ as *minimal* if the following conditions are satisfied:

1. $\mathcal{I}_{\mathcal{A}}$ is a countermodel;
2. $\mathcal{I}_{\mathcal{A}^{-a}} \models \mathcal{T}$; and
3. $\mathcal{I}_{\mathcal{A}^{-r(a,b)}} \models \mathcal{T}$.

The following lemma shows that the size of a minimal ABox is polynomial in $|\mathcal{T}|$.

Lemma 47 *Let $\mathcal{I}_{\mathcal{A}}$ be the canonical model of a minimal ABox $\mathcal{A}$. Then, $|\Delta^{\mathcal{I}_{\mathcal{A}}}| \leq |\mathcal{T}|$.*

Proof. By Condition 1 of minimal ABoxes, $\mathcal{I}_{\mathcal{A}}$ is a countermodel. So $\mathcal{I}_{\mathcal{A}} \not\models \mathcal{T}$. Then there is $C \sqsubseteq A \in \mathcal{T}$ such that $a \in (C \setminus A)^{\mathcal{I}_{\mathcal{A}}}$, for some $a \in \Delta^{\mathcal{I}_{\mathcal{A}}}$. If $a \in C^{\mathcal{I}_{\mathcal{A}}}$ then (by Lemma 11) there is a homomorphism $h : \mathcal{I}_C \to \mathcal{I}_{\mathcal{A}}$ mapping ρ_C to a, where ρ_C is the root of $\mathcal{I}_C$. We need to show that h is surjective. Suppose this is not the case. Then, there is $d \in \Delta^{\mathcal{I}_{\mathcal{A}}}$ such that $d \notin \mathsf{Im_h}$, where $\mathsf{Im_h} = \{e \in \Delta^{\mathcal{I}_{\mathcal{A}}} \mid e = h(p) \text{ for some } p \in \Delta^{\mathcal{I}_C}\}$. Now, denote as $\mathcal{I}_{\mathcal{A}^{-d}}$ the result of removing $d \notin \mathsf{Im_h}$ from $\mathcal{I}_{\mathcal{A}}$. Since $\mathcal{I}_{\mathcal{A}^{-d}}$ is a subinterpretation of $\mathcal{I}_{\mathcal{A}}$, if $a \notin A^{\mathcal{I}_{\mathcal{A}}}$ then $a \notin A^{\mathcal{I}_{\mathcal{A}^{-d}}}$. So $a \in (C \setminus A)^{\mathcal{I}_{\mathcal{A}^{-d}}}$, which means that $\mathcal{I}_{\mathcal{A}^{-d}} \not\models \mathcal{T}$. This contradicts the fact that $\mathcal{I}_{\mathcal{A}^{-d}} \models \mathcal{T}$ for any element d from $\Delta^{\mathcal{I}_{\mathcal{A}}}$ (Condition 3 of minimal ABoxes). Since $C \sqsubseteq A \in \mathcal{T}$, we know that $|\Delta^{\mathcal{I}_C}| \leq |\mathcal{T}|$. Thus, $|\Delta^{\mathcal{I}_{\mathcal{A}}}| \leq |\Delta^{\mathcal{I}_C}| \leq |\mathcal{T}|$. ❑

Algorithm 2 minimizes $\mathcal{A}$ to an $\mathcal{A}'$ with the properties described above. Since the Algorithm receives a positive counterexample, we know that $\mathcal{I}_{\mathcal{A}}$ is not a model of $\mathcal{T}$, that is, $\mathcal{I}_{\mathcal{A}} \not\models \mathcal{T}$. In order to satisfy Condition 1 above and reduce $\mathcal{A}$ (Conditions 2 and 3), Algorithm 2 applies rules 'Concept saturate', 'Domain Minimize' and 'Role Minimize'.

We now show Lemma 48, which implies Lemma 5.

Lemma 48 *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 in polynomial time 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.*

Proof. By Lemma 47, if $\mathcal{A}$ is a minimal ABox then $|\mathsf{Ind}(\mathcal{A})| \leq |\mathcal{T}|$. Then, we only need the following claims to show this lemma.

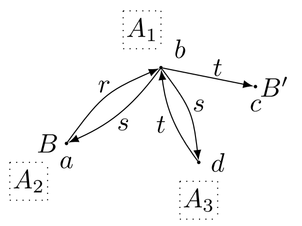
Claim 1. Algorithm 2 computes a minimal ABox $\mathcal{A}$.

For Condition 1, we have that, in Line 2, Algorithm 2 concept saturates $\mathcal{A}$ with $\mathcal{H}$. Then, after computing Line 2, we have that $\mathcal{I}_{\mathcal{A}} \models \mathcal{H}$. Since the minimization rules described above do not remove any concept name implied by $\mathcal{H}$, the ABox computed by the algorithm is a model of $\mathcal{H}$ in all steps that follow Line 2. By definition of the rules, at least one example $(\mathcal{A}, A(a))$ is such that $(\mathcal{T}, \mathcal{A}) \models A(a)$ and $(\mathcal{H}, \mathcal{A}) \not\models A(a)$, which is the counterexample where the rules are being applied. So for all iterations of Algorithm 2, $\mathcal{I}_{\mathcal{A}} \not\models \mathcal{T}$.

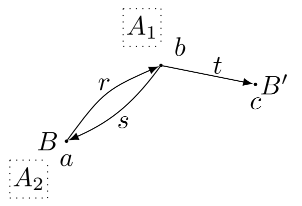
For Condition 2, suppose there is $\mathcal{A}^{-d}$ such that $\mathcal{I}_{\mathcal{A}^{-d}} \not\models \mathcal{T}$. Then there is $C \sqsubseteq A \in \mathcal{T}$ and $a \in \Delta^{\mathcal{I}_{\mathcal{A}^{-d}}}$ such that $a \in (C \setminus A)^{\mathcal{I}_{\mathcal{A}^{-d}}}$. This contradicts the fact that, in Line 5, domain minimization was applied in $\mathcal{A}$ for all counterexamples. Thus, $\mathcal{I}_{\mathcal{A}^{-d}} \models \mathcal{T}$. The argument is similar for role minimization (Condition 3).

Claim 2. Algorithm 2 runs in polynomially many steps with respect to $|\mathcal{A}|$ and $|\mathsf{N_C} \cap \Sigma_{\mathcal{T}}|$, where $\mathsf{N_C} \cap \Sigma_{\mathcal{T}}$ are the concept names in the vocabulary.

We know that the number of possible concept name assertions in $\mathcal{A}$ is $|\mathsf{N_C} \cap \Sigma_{\mathcal{T}}| \cdot |\mathsf{Ind}(\mathcal{A})|$. So, in Line 2, the number of applications of the rule Concept Saturate with $\mathcal{H}$ is bounded by $|\mathsf{N_C} \cap \Sigma_{\mathcal{T}}| \cdot |\mathsf{Ind}(\mathcal{A})|$. Also, the number of iterations in Line 3 is at most $|\mathsf{N_C} \cap \Sigma_{\mathcal{T}}| \cdot |\mathsf{Ind}(\mathcal{A})|$. Since role and domain minimization is linear in $|\mathcal{A}|$, we have that Algorithm 2 is polynomial with respect to $|\mathcal{A}|$ and $|\mathsf{N_C} \cap \Sigma_{\mathcal{T}}|$. ❑



(a) Initial $\mathcal{A}$ with the AQs $A_1(b), A_2(a), A_3(d)$



(b) Minimal $\mathcal{A}$ with the AQs $A_1(b), A_2(a)$

Figure 7: Minimizing $\mathcal{A}$

As an example, let $\mathcal{T} = \{\exists s.\exists r.(\exists s.B \sqcap \exists t.B') \sqsubseteq A_1, \exists r.(\exists t.B' \sqcap \exists s.B) \sqsubseteq A_2, \exists t.(\exists s.B \sqcap \exists t.B') \sqsubseteq A_3, \exists r.\exists t.B' \sqsubseteq B\}$, $\mathcal{H} = \{\exists r.\exists t.B' \sqsubseteq B\}$ and $\mathcal{A} = \{r(a,b), s(b,a), t(b,c),\ s(b,d), t(d,b), B'(c)\}$. After concept saturating $\mathcal{A}$ with $\mathcal{H}$ we have that $B(a)$ is added to $\mathcal{A}$. Then we have $(\mathcal{A}, A_1(b))$, $(\mathcal{A}, A_2(a))$ and $(\mathcal{A}, A_3(d))$ as counterexamples (Figure 7-(a)). Assume Algorithm 2 starts minimizing $\mathcal{A}$ with the counterexample $(\mathcal{A}, A_1(b))$. The algorithm eliminates $s(b,d)$ and $t(d,b)$ from $\mathcal{A}$. As a result, $(\mathcal{A}, A_3(d))$ is not a counterexample any more. In the next iteration, the algorithm tries to minimize $\mathcal{A}$ with $A_2(a)$, which does not eliminates any other assertion from $\mathcal{A}$. So $\mathcal{A}$ is now minimal. The result of minimizing $\mathcal{A}$ is shown by Figure 7-(b), it contains now only $(\mathcal{A}, A_1(b))$ and $(\mathcal{A}, A_2(a))$ as counterexamples.

We have seen that a minimal ABox $\mathcal{A}$ is a countermodel bounded by $|\mathcal{T}|$. Algorithm 3 is based on two operations (i) minimization, presented above, and (ii) *cycle unfolding*. The cycle unfolding operation doubles the length of a cycle in $\mathcal{A}$. By increasing the length of cycles and then minimizing, the algorithm proceeds unfolding elements until $\mathcal{A}$ is tree shaped. We say that $\mathcal{A}$ has a (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 $0 \leq i < k$. For a cycle $c = a_0 \cdot r_1 \cdot a_1 \cdot ... \cdot r_k \cdot a_k$, denote as $\mathsf{nodes}(c) = \{a_0, a_1, ..., a_{k-1}\}$ the set of individuals that occur in c. Also, $\mathsf{roles}(c) = \{r_1, r_2, ..., r_k\}$ is the set of roles that occur in c. We denote by $\widehat{a}$ the copy of an element a created by the unfolding cycle operation described below. The set of copies of individuals that occur in c is denoted by $\mathsf{nodes}(\widehat{c}) = \{\widehat{a_0}, \widehat{a_1}, ..., \widehat{a}_{k-1}\}$. Let $\mathcal{I}_\mathcal{A}$ be the canonical interpretation of an ABox $\mathcal{A}$. An element $a \in \Delta^{\mathcal{I}_\mathcal{A}}$ is *folded* if there is a cycle $c = a_0 \cdot r_1 \cdot a_1 \cdot ... \cdot r_k \cdot a_k$ with $a = a_0 = a_k$. Without loss of generality we assume that $r_1(a_0, a_1) \in \mathcal{A}$. The *cycle unfolding* of c is described below.

1. We first open the cycle by removing $r_1(a_0, a_1)$ from $\mathcal{A}$. So $r_1^{\mathcal{I}_\mathcal{A}} := r_1^{\mathcal{I}_\mathcal{A}} \setminus \{(a_0, a_1)\}$.
2. Then we create copies of the nodes in the cycle:
   - $\Delta^{\mathcal{I}_\mathcal{A}} := \Delta^{\mathcal{I}_\mathcal{A}} \cup \{\widehat{b} \mid b \in \mathsf{nodes}(c)\}$
   - $A^{\mathcal{I}_\mathcal{A}} := A^{\mathcal{I}_\mathcal{A}} \cup \{\widehat{b} \mid b \in A^{\mathcal{I}_\mathcal{A}}\}$
   - $r^{\mathcal{I}_\mathcal{A}} := r^{\mathcal{I}_\mathcal{A}} \cup \{(\widehat{b}, \widehat{d}) \mid (b, d) \in r^{\mathcal{I}_\mathcal{A}}\} \cup \{(\widehat{b}, e) \mid (b, e) \in r^{\mathcal{I}_\mathcal{A}}, e \notin \mathsf{nodes}(c)\}$
3. As a third step we close again the cycle, now with double size. So we update $r_1^{\mathcal{I}_\mathcal{A}} := r_1^{\mathcal{I}_\mathcal{A}} \cup \{(a_0, \widehat{a_1}), (\widehat{a_0}, a_1)\}$.

We now show that our cycle unfolding operation maintains the invariant that if $(\mathcal{A}, A(a))$ is a counterexample for $\mathcal{T}$ relative to $\mathcal{H}$ then $(\mathcal{A}, A(a))$ will remain as a counterexample after applying this operation over an arbitrary cycle in $\mathcal{A}$. This is obtained by Lemmas 49 and 50.

Lemma 49 *Let $\mathcal{A}'$ be the result of unfolding a cycle c in $\mathcal{A}$. Then the following relation $S \subseteq \Delta^{\mathcal{I}_\mathcal{A}} \times \Delta^{\mathcal{I}_{\mathcal{A}'}}$ is a simulation $\mathcal{I}_\mathcal{A} \Rightarrow \mathcal{I}_{\mathcal{A}'}$:*

- *for $a \in \Delta^{\mathcal{I}_\mathcal{A}} \setminus \mathsf{nodes}(c)$, $(a^{\mathcal{I}_\mathcal{A}}, a^{\mathcal{I}_{\mathcal{A}'}}) \in S$;*
- *for $a \in \mathsf{nodes}(c)$, $(a^{\mathcal{I}_\mathcal{A}}, a^{\mathcal{I}_{\mathcal{A}'}}) \in S$ and $(a^{\mathcal{I}_\mathcal{A}}, \widehat{a}^{\mathcal{I}_{\mathcal{A}'}}) \in S$.*

Proof. We need to show that S is a simulation $\mathcal{I}_\mathcal{A} \Rightarrow \mathcal{I}_{\mathcal{A}'}$. That is, for $d, d_1 \in \Delta^{\mathcal{I}_\mathcal{A}}$ and $e, e_1 \in \Delta^{\mathcal{I}_{\mathcal{A}'}}$:

1. for all concept names $A \in \mathsf{N_C}$ and all $(d, e) \in S$, if $d \in A^{\mathcal{I}_\mathcal{A}}$ then $e \in A^{\mathcal{I}_{\mathcal{A}'}}$;
2. for all role names $r \in \mathsf{N_R}$, all $(d_1, e_1) \in S$ and all $d_2 \in \Delta^{\mathcal{I}_\mathcal{A}}$, if $(d_1, d_2) \in r^{\mathcal{I}_\mathcal{A}}$ then there exists $e_2 \in \Delta^{\mathcal{I}_{\mathcal{A}'}}$ such that $(e_1, e_2) \in r^{\mathcal{I}_{\mathcal{A}'}}$ and $(d_2, e_2) \in S$.

For Point 1 we have that by definition of the cycle unfolding operation (Step 2), if $a \in A^{\mathcal{I}_\mathcal{A}}$ then $a \in A^{\mathcal{I}_{\mathcal{A}'}}$ and $\widehat{a} \in A^{\mathcal{I}_{\mathcal{A}'}}$. Point 2 follows from Claims 1 and 2 below.

Claim 1. If $a^{\mathcal{I}_\mathcal{A}}$ has an r-successor b then $a^{\mathcal{I}_{\mathcal{A}'}}$ has an r-successor d with $(b, d) \in S$.

By definition of the cycle unfolding operation (Step 1), $r_1(a_0, a_1)$ is the only role assertion removed from $\mathcal{A}$. In Step 3 we include $(a_0, \widehat{a_1})$ to $r_1^{\mathcal{I}_\mathcal{A}}$. By definition of S, we have that $(a_1, \widehat{a_1}) \in S$. So if $a^{\mathcal{I}_\mathcal{A}}$ has an r-successor b then $a^{\mathcal{I}_{\mathcal{A}'}}$ has an r-successor d with $(b, d) \in S$.

Claim 2. If $a^{\mathcal{I}_\mathcal{A}}$ has an r-successor b then $\widehat{a}^{\mathcal{I}_{\mathcal{A}'}}$ has an r-successor e with $(b, e) \in S$.

For $(a_0, a_1) \in r_1^{\mathcal{I}_\mathcal{A}}$, in Step 3 we include $(\widehat{a}_0, a_1)$ to $r_1^{\mathcal{I}'_\mathcal{A}}$. By definition of S, $(a_1^{\mathcal{I}_{\mathcal{A}'}}, a_1^{\mathcal{I}_{\mathcal{A}'}}) \in S$. Otherwise, in Step 2 we have that for all r-successors $b^{\mathcal{I}_\mathcal{A}}$ of $a^{\mathcal{I}_\mathcal{A}}$ such that $b \notin \mathsf{nodes}(c)$, $(\widehat{a}^{\mathcal{I}_{\mathcal{A}'}}, b^{\mathcal{I}_{\mathcal{A}'}}) \in r^{\mathcal{I}_{\mathcal{A}'}}$. By definition of S, $(b^{\mathcal{I}_\mathcal{A}}, b^{\mathcal{I}_{\mathcal{A}'}}) \in S$. Also, in Step 2, for the r-successors $b^{\mathcal{I}_\mathcal{A}}$ of $a^{\mathcal{I}_\mathcal{A}}$ such that $b \in \mathsf{nodes}(c)$, we have that $(\widehat{a}^{\mathcal{I}_{\mathcal{A}'}}, \widehat{b}^{\mathcal{I}_{\mathcal{A}'}}) \in r^{\mathcal{I}_{\mathcal{A}'}}$, where $\widehat{b}^{\mathcal{I}_{\mathcal{A}'}}$ is the copy of $b^{\mathcal{I}_{\mathcal{A}'}}$. Again, by definition of S, $(b^{\mathcal{I}_\mathcal{A}}, \widehat{b}^{\mathcal{I}_{\mathcal{A}'}}) \in S$. ❏

Lemma 50 *Let $\mathcal{A}'$ be the result of unfolding a cycle c in $\mathcal{A}$. Let $h_* : \mathcal{I}_{\mathcal{A}'} \to \mathcal{I}_\mathcal{A}$ be the following mapping:*

- *for $a \in \Delta^{\mathcal{I}_\mathcal{A}} \setminus \mathsf{nodes}(c)$, $h_*(a^{\mathcal{I}_{\mathcal{A}'}}) = a^{\mathcal{I}_\mathcal{A}}$;*
- *for $a \in \mathsf{nodes}(c)$, $h_*(a^{\mathcal{I}_{\mathcal{A}'}}) = a^{\mathcal{I}_\mathcal{A}}$ and $h_*(\widehat{a}^{\mathcal{I}_{\mathcal{A}'}}) = a^{\mathcal{I}_\mathcal{A}}$.*

Then, $h_* : \mathcal{I}_{\mathcal{A}'} \to \mathcal{I}_\mathcal{A}$ is a homomorphism.

Proof. By definition of the cycle unfolding operation, no concept name assertion is removed from $\mathcal{A}'$. So $a \in A^{\mathcal{I}_{\mathcal{A}'}}$ iff $a \in A^{\mathcal{I}_\mathcal{A}}$. Also, in Step 2 of the cycle unfolding operation we have that $\widehat{a} \in A^{\mathcal{I}_{\mathcal{A}'}}$ iff $a \in A^{\mathcal{I}_\mathcal{A}}$. So if $a \in A^{\mathcal{I}_{\mathcal{A}'}}$ or $\widehat{a} \in A^{\mathcal{I}_{\mathcal{A}'}}$ then $h_*(a) = h_*(\widehat{a}) = a \in A^{\mathcal{I}_\mathcal{A}}$. Now, for $(a, b) \in r^{\mathcal{I}_{\mathcal{A}'}}$, we make a case distinction:

- $a, b \notin \mathsf{nodes}(\widehat{c})$: in this case, the cycle unfolding operation does not include any new role assertion. Then, $(a, b) \in r^{\mathcal{I}_{\mathcal{A}'}}$ implies $(a, b) \in r^{\mathcal{I}_\mathcal{A}}$.
- $\widehat{a}, \widehat{b} \in \mathsf{nodes}(\widehat{c})$: by Step 2 of the cycle unfolding operation, if $(\widehat{a}, \widehat{b}) \in r^{\mathcal{I}_{\mathcal{A}'}}$ then $(a, b) \in r^{\mathcal{I}_\mathcal{A}}$.
- $\widehat{a} \in \mathsf{nodes}(\widehat{c})$ and $b \notin \mathsf{nodes}(\widehat{c})$: for $(\widehat{a}_0, a_1) \in r_1^{\mathcal{I}_{\mathcal{A}'}}$ we know that $(a_0, a_1) \in r_1^{\mathcal{I}_\mathcal{A}}$. Otherwise, by Step 2, if $(\widehat{a}, b) \in r^{\mathcal{I}_{\mathcal{A}'}}$ then $(a, b) \in r^{\mathcal{I}_\mathcal{A}}$.
- $a \notin \mathsf{nodes}(\widehat{c})$ and $\widehat{b} \in \mathsf{nodes}(\widehat{c})$: by the definition of the cycle unfolding operation there is only one case, in Step 3, which is $(a_0, \widehat{a}_1) \in r_1^{\mathcal{I}_{\mathcal{A}'}}$. In this case we know that $(a_0, a_1) \in r_1^{\mathcal{I}_\mathcal{A}}$.

In all cases we have that for $a, b \in \Delta^{\mathcal{I}_{\mathcal{A}'}}$, $(a, b) \in r^{\mathcal{I}_{\mathcal{A}'}}$ implies $(h_*(a), h_*(b)) \in r^{\mathcal{I}_\mathcal{A}}$. ❏

Before we show Lemma 6 we need the following lemma, which shows the progress of our cycle unfolding operations.

Lemma 51 *Let $\mathcal{I}_n$ be the canonical model of the minimal ABox computed in the n-th iteration in Line 5 of Algorithm 3. Assume $\mathcal{I}_n$ has a cycle. For all $n \geq 0$, $|\Delta^{\mathcal{I}_{n+1}}| > |\Delta^{\mathcal{I}_n}|$.*

Proof. By assumption $\mathcal{I}_n$ has a cycle c. Let $\mathcal{I}'_n$ be the result of unfolding cycle c and $\mathcal{I}_{n+1}$ be the result of minimizing $\mathcal{I}'_n$. Let $h_* : \mathcal{I}'_n \to \mathcal{I}_n$ be the homomorphism defined in Lemma 50. Let $g = h_*|_{\Delta^{\mathcal{I}_{n+1}}}$ be h_* restricted to $\Delta^{\mathcal{I}_{n+1}} \subseteq \Delta^{\mathcal{I}'_n}$. Since $\mathcal{I}_{n+1}$ is a subinterpretation of $\mathcal{I}'_n$, $g : \mathcal{I}_{n+1} \to \mathcal{I}_n$ is a homomorphism.

Claim 1. $g : \mathcal{I}_{n+1} \rightarrow \mathcal{I}_n$ is a surjective homomorphism.

Suppose g is not surjective. Since $\mathcal{I}_{n+1}$ is a countermodel (Condition 1 of minimal ABoxes) there is $C \sqsubseteq A \in \mathcal{T}$ such that $a \in (C \setminus A)^{\mathcal{I}_{n+1}}$, with $a \in \Delta^{\mathcal{I}_{n+1}}$. Let $\mathcal{J}$ be the subinterpretation of $\mathcal{I}_n$ determined by the range of g. By the cycle unfolding definition, $a \in A^{\mathcal{I}_{n+1}}$ iff $g(a) \in A^{\mathcal{I}_n}$. Then $g(a) \in (C \setminus A)^{\mathcal{J}}$. Since $\mathcal{I}_n$ is the canonical model of a minimal ABox, if g is not surjective then this contradicts Condition 2 of minimal ABoxes.

Claim 2. Suppose $g : \mathcal{I}_{n+1} \rightarrow \mathcal{I}_n$ is an injective homomorphism. Then, for $d_1, d_2 \in \Delta^{\mathcal{I}_{n+1}}$, $(g(d_1), g(d_2)) \in r^{\mathcal{I}_n}$ implies $(d_1, d_2) \in r^{\mathcal{I}_{n+1}}$.

Suppose this is not the case and there is $d_1, d_2 \in \Delta^{\mathcal{I}_{n+1}}$ such that $(g(d_1), g(d_2)) \in r^{\mathcal{I}_n}$ and $(d_1, d_2) \notin r^{\mathcal{I}_{n+1}}$. Let $\mathcal{J}$ be the result of removing $(g(d_1), g(d_2))$ from $r^{\mathcal{I}_n}$. Since g is injective, $g : \mathcal{I}_{n+1} \rightarrow \mathcal{J}$ is also a homomorphism. As $\mathcal{I}_{n+1}$ is a countermodel (Condition 1 of minimal ABoxes) there is $C \sqsubseteq A \in \mathcal{T}$ such that $a \in (C \setminus A)^{\mathcal{I}_{n+1}}$, with $a \in \Delta^{\mathcal{I}_{n+1}}$. Then, $g(a) \in C^{\mathcal{J}}$. By the cycle unfolding definition, $a \in A^{\mathcal{I}_{n+1}}$ iff $g(a) \in A^{\mathcal{I}_n}$. By definition of $\mathcal{J}$, $g(a) \in A^{\mathcal{I}_n}$ iff $g(a) \in A^{\mathcal{J}}$. Then $g(a) \in (C \setminus A)^{\mathcal{J}}$. Since $\mathcal{J}$ is $\mathcal{I}_n$ with $(g(d_1), g(d_2))$ removed from $r^{\mathcal{I}_n}$, this contradicts the fact that $\mathcal{I}_n$ is role minimal (Condition 3 of minimal ABoxes).

Claim 3. $g : \mathcal{I}_{n+1} \rightarrow \mathcal{I}_n$ is not an injective homomorphism.

Recall that cycle c is a sequence $a_0 \cdot r_1 \cdot a_1 \cdot ... \cdot r_k \cdot a_k$, with $a_0 = a_k$, where we defined w.l.g. that $(a_0, a_1) \in r_1^{\mathcal{I}_n}$. As g is surjective (Claim 1), for all $0 \leq i \leq k$, a_i or $\widehat{a}_i$ is in $\Delta^{\mathcal{I}_{n+1}}$. Suppose to the contrary that g is injective. Then,

$(*)$ for all $0 \leq i \leq k$, exactly one of $\{a_i, \widehat{a}_i\}$ is in $\Delta^{\mathcal{I}_{n+1}}$.

Assume that $a_0 \in \Delta^{\mathcal{I}_{n+1}}$ (the case where $\widehat{a}_0 \in \Delta^{\mathcal{I}_{n+1}}$ is analogous). By Point 1 of the definition of cycle unfolding, $(a_0, a_1) \notin r_1^{\mathcal{I}'_n}$, where $\mathcal{I}'_n$ is the result of unfolding a cycle in $\mathcal{I}_n$. Then $(a_0, a_1) \notin r_1^{\mathcal{I}_{n+1}}$. As $(g(a_0), g(a_1)) \in r_1^{\mathcal{I}_n}$, if $a_1 \in \Delta^{\mathcal{I}_{n+1}}$ then this contradicts Claim 2. So $a_1 \notin \Delta^{\mathcal{I}_{n+1}}$. For $k = 1$ (that is, the cycle is a reflexive element) we have $a_0 = a_1$. This contradicts our assumption that $a_0 \in \Delta^{\mathcal{I}_{n+1}}$. Now suppose $k > 1$. As $a_1 \notin \Delta^{\mathcal{I}_{n+1}}$, by $(*)$, $\widehat{a}_1 \in \Delta^{\mathcal{I}_{n+1}}$. By Points 2 and 3 of the definition of cycle unfolding, $r_1(a_0, \widehat{a}_1)$, $r_1(\widehat{a}_0, a_1)$ are the only role assertions between elements in $\mathsf{nodes}(c)$ and $\mathsf{nodes}(\widehat{c})$ in $\mathcal{I}'_n$. This means that

$(*')$ neither $(\widehat{a}_i, a_{i+1})$ or $(a_{i+1}, \widehat{a}_i)$ are in $r_{i+1}^{\mathcal{I}_{n+1}} \subseteq r_{i+1}^{\mathcal{I}'_n}$, for $1 \leq i < k$.

Then, for $1 \leq i < k$, $a_i \notin \Delta^{\mathcal{I}_{n+1}}$ and $\widehat{a}_i \in \Delta^{\mathcal{I}_{n+1}}$, otherwise we would obtain a contradiction with Claim 2 or $(*)$. In particular, $a_{k-1} \notin \Delta^{\mathcal{I}_{n+1}}$ and $\widehat{a}_{k-1} \in \Delta^{\mathcal{I}_{n+1}}$. Since $a_0 = a_k \in \Delta^{\mathcal{I}_{n+1}}$, by $(*)$, we have $\widehat{a}_0 = \widehat{a}_k \notin \Delta^{\mathcal{I}_{n+1}}$. By definition of c, either (a_{k-1}, a_k) or (a_k, a_{k-1}) are in $r_k^{\mathcal{I}_n}$. Together with the fact $(*')$ that neither $(\widehat{a}_{k-1}, a_k)$ or $(a_k, \widehat{a}_{k-1})$ are in $r_k^{\mathcal{I}_{n+1}} \subseteq r_k^{\mathcal{I}'_n}$, we obtain a contradiction with Claim 2. Then g is not injective. Since g is surjective (Claim 1) and not injective (Claim 3), $|\Delta^{\mathcal{I}_{n+1}}| > |\Delta^{\mathcal{I}_n}|$. ❑

We show Lemma 52, which implies Lemma 6.

Lemma 52 *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 in polynomial time 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.*

Proof. The fact that the computed ABox is tree shaped follows from Line 3. Also, by Lemma 5 the size of the ABox is bounded by $|\mathcal{T}|$. So it remains to show that Algorithm 3 terminates after at polynomially many steps in $|\mathcal{T}|$ and $|\mathcal{A}|$. By Lemma 5, Lines 2 and 5 is polynomial in $|\mathcal{A}|$ and $|\mathsf{N_C} \cap \Sigma_{\mathcal{T}}|$. Also, unfolding a cycle c in Line 4 is linear in $|\mathcal{A}|$. It remains to show that the number of iterations is bounded by $|\mathcal{T}|$. Let $\mathcal{I}_n$ be the minimal ABox computed in the n-th iteration in Line 5 of Algorithm 3. By Lemma 5, for all n iterations of Algorithm 3, in Line 5 $|\Delta^{\mathcal{I}_n}|$ is bounded by $|\mathcal{T}|$. By Lemma 51, after each $n+1$-th iteration of the algorithm, $|\Delta^{\mathcal{I}_{n+1}}|$ increases by at least one element with respect to $|\Delta^{\mathcal{I}_n}|$. So the number of iterations is bounded by $|\mathcal{T}|$. ❑

We now detail the polynomials of Definition 45 involved in our proof for $\mathcal{ELH}_{\mathsf{lhs}}$. Let $\mathcal{T}$ be the target $\mathcal{ELH}_{\mathsf{lhs}}$ TBox with signature $\Sigma_{\mathcal{T}}$. We want to define $p_1(\cdot), p_2(\cdot,\cdot), p_3(\cdot,\cdot)$ such that:

- $p_1(|\mathcal{T}|)$ is the polynomial bound on $|\mathcal{H}|$;
- $p_2(|C \sqsubseteq A|)$ is the polynomial time bound for computing $f_{\mathsf{MEM}}(C \sqsubseteq A)$;
- $p_3(|\mathcal{T}|, |(\mathcal{A}, B(a))|)$ is the polynomial time bound for computing $f_{\mathsf{EQ}}(\mathcal{T}, \mathcal{H}, (\mathcal{A}, B(a)))$ (where $B \in \mathsf{N_C}$).

In the learning algorithm for $\mathcal{ELH}_{\mathsf{lhs}}$ proved in this paper, $\mathcal{H}$ is the union of $\mathcal{H}_{add}$ and $\mathcal{H}_{basic}$. We have that $|\mathcal{H}_{basic}|$ is bounded by $|\Sigma_{\mathcal{T}}|^2$ and $\mathcal{H}_{add}$ has at most $|\mathcal{T}| \cdot |\Sigma_{\mathcal{T}}|$ CIs. The right hand side of each CI in $\mathcal{H}_{add}$ has size 1 and left hand side of each CI is bounded by $|\Sigma_{\mathcal{T}}| \cdot |\mathcal{T}|$. Then, $|\mathcal{H}_{add}| \leq |\mathcal{T}| \cdot |\Sigma_{\mathcal{T}}| \cdot (|\Sigma_{\mathcal{T}}| \cdot |\mathcal{T}| + 1)$. As $|\Sigma_{\mathcal{T}}| \leq |\mathcal{T}|$, we have $|\mathcal{H}| \leq p_1(|\mathcal{T}|)$ with $p_1(|\mathcal{T}|) = |\mathcal{T}|^4 + 2 \cdot |\mathcal{T}|^2$ (we note that normally the size of the signature is much smaller than the size of the TBox and, so, this rough estimate only serves for the purpose of defining our polynomials). As $f_{\mathsf{MEM}}(C \sqsubseteq A) = (\{C(a)\}, A(a))$, we have $p_2(|C \sqsubseteq A|) = |C| + 1$. By the proof of the lemma above, we have that $p_3(|\mathcal{T}|, |(\mathcal{A}, B(a))|) =$

$$k \cdot \underbrace{|\mathcal{T}|}_{\text{unfold/min times}} \left(\overbrace{|\mathcal{A}|}^{\text{unfolding}} + \overbrace{p_1(|\mathcal{T}|)|\mathcal{A}|}^{\text{concept sat. with } \mathcal{H}} + \underbrace{p_1(|\mathcal{T}|)|\mathcal{T}||\mathcal{A}|^2}_{\text{domain and role min.}} \right)$$

for some $k \in \mathbb{N}$. That is, $p_3(|\mathcal{T}|, |(\mathcal{A}, B(a))|)$ is $O(|\mathcal{T}|^6 \cdot |\mathcal{A}|^2))$.

Remaining Positive Results in Table 1

We proved that

- the learning framework $\mathfrak{F}_{\mathcal{D}}(\text{DL-Lite}^{\exists}_{\mathcal{H}}, \mathcal{ELI}\text{-IQ})$ is polynomial query exact learnable and

- the learning framework $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{lhs}}, \text{AQ})$ is polynomial time exact learnable.

Now we discuss how one can obtain the remaining positive results for the data retrieval framework presented in Table 1 for polynomial *time* exact learnability.

(1) $\mathfrak{F}_\mathcal{D}(\mathcal{EL}_{\mathsf{rhs}}, \mathcal{ELI}\text{-IQ})$ and $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{rhs}}, \mathcal{ELI}\text{-IQ})$.

Observe that given an $\mathcal{ELI}$-IQ $C(a)$ and $\mathcal{ELH}$ TBox $\mathcal{H}$, the problem of checking whether $(\mathcal{H}, \mathcal{A}) \models C(a)$ is in PTime in combined complexity (Bienvenu et al. 2013). (This is in contrast to the NP-completeness of this problem for DL-Lite$^\exists_\mathcal{H}$ TBoxes (Kikot, Kontchakov, and Zakharyaschev 2011).) Now the polynomial query learning algorithm for the framework $\mathfrak{F}_\mathcal{D}(\text{DL-Lite}^\exists_\mathcal{H}, \mathcal{ELI}\text{-IQ})$ given above is (modulo minor modifications) a polynomial *time* learning algorithm for the frameworks $\mathfrak{F}_\mathcal{D}(\mathcal{EL}_{\mathsf{rhs}}, \mathcal{ELI}\text{-IQ})$ and $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{rhs}}, \mathcal{ELI}\text{-IQ})$: the checks '$(\mathcal{H}, \mathcal{A}) \models C(a)$' can be performed in polynomial time, membership queries coincide, and a close inspection of the learning algorithm shows that if the target is an $\mathcal{EL}_{\mathsf{rhs}}$ or $\mathcal{ELH}_{\mathsf{rhs}}$ TBox, then all equivalence queries contain $\mathcal{EL}_{\mathsf{rhs}}$ or, respectively, $\mathcal{ELH}_{\mathsf{rhs}}$ TBoxes only.

(2) $\mathfrak{F}_\mathcal{D}(\mathcal{EL}_{\mathsf{rhs}}, \mathcal{EL}\text{-IQ})$ and $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{rhs}}, \mathcal{EL}\text{-IQ})$.

For $\mathfrak{F}_\mathcal{D}(\mathcal{EL}_{\mathsf{rhs}}, \mathcal{EL}\text{-IQ})$ and $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{rhs}}, \mathcal{EL}\text{-IQ})$ polynomial time learnability follow from polynomial time learnability of $\mathfrak{F}_\mathcal{D}(\mathcal{EL}_{\mathsf{rhs}}, \mathcal{ELI}\text{-IQ})$ and $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{rhs}}, \mathcal{ELI}\text{-IQ})$. In fact, removing parent/child merging from the learning algorithms for $\mathfrak{F}_\mathcal{D}(\mathcal{EL}_{\mathsf{rhs}}, \mathcal{ELI}\text{-IQ})$ and $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{rhs}}, \mathcal{ELI}\text{-IQ})$ gives the required learning algorithms.

(3) $\mathfrak{F}_\mathcal{D}(L, Q)$ for $L \in \{\mathcal{EL}_{\mathsf{lhs}}, \mathcal{ELH}_{\mathsf{lhs}}\}$ and $Q \in \{\text{AQ}, \mathcal{EL}\text{-IQ}, \mathcal{ELI}\text{-IQ}, \text{CQ}\}$.

We focus on $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{lhs}}, \text{CQ})$ and reduce learning $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{lhs}}, \text{CQ})$ to learning $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{lhs}}, \text{AQ})$ (and then use that polynomial time learnability of $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{lhs}}, \text{AQ})$ has been proved above). The remaining learning frameworks are considered similarly.

To learn $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}_{\mathsf{lhs}}, \text{CQ})$, we can assume that the RIs of the target TBox $\mathcal{T}$ are known by the learner. The following lemma shows that given a positive counterexample $(\mathcal{A}, q)$ with q a CQ, we can find a positive counterexample with an AQ by posing membership queries of the form $(\mathcal{T}, \mathcal{A}) \models A(b)$ to the oracle, with $A \in \mathsf{N_C} \cap \Sigma_\mathcal{T}$ and $b \in \mathsf{Ind}(\mathcal{A})$.

Lemma 53 *For $\mathcal{T}$ an $\mathcal{ELH}_{\mathsf{lhs}}$ target TBox and $\mathcal{H}$ an $\mathcal{ELH}_{\mathsf{lhs}}$ hypothesis TBox, assume that $\mathcal{H} \models r \sqsubseteq s$ iff $\mathcal{T} \models r \sqsubseteq s$, for all $r, s \in \mathsf{N_R}$. If $(\mathcal{A}, q)$ is a positive counterexample then there exists a concept name A and an individual $b \in \mathsf{Ind}(\mathcal{A})$ such that $(\mathcal{A}, A(b))$ is also a positive counterexample.*

Proof. As $(\mathcal{A}, q)$ is a positive counterexample, $(\mathcal{T}, \mathcal{A}) \models q$ and $(\mathcal{H}, \mathcal{A}) \not\models q$. Then, by Lemma 13, $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q$ and $\mathcal{I}_{\mathcal{H},\mathcal{A}} \not\models q$, where $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ and $\mathcal{I}_{\mathcal{H},\mathcal{A}}$ are the canonical models of $(\mathcal{T}, \mathcal{A})$ and $(\mathcal{H}, \mathcal{A})$, respectively. Let π be a mapping from the terms of q into $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi(a) = a$ for all individuals a in q and such that

- if $r(t_1, t_2)$ is a conjunct in q, then $(\pi(t_1), \pi(t_2)) \in r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$;
- if $A(t)$ is a conjunct of q, then $\pi(t) \in A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

We have $(\pi(t_1), \pi(t_2)) \in r^{\mathcal{I}_{\mathcal{H},\mathcal{A}}}$ for all conjuncts $r(t_1, t_2)$ of q since $\mathcal{H} \models r \sqsubseteq s$ iff $\mathcal{T} \models r \sqsubseteq s$ for all $r, s \in \mathsf{N_R}$. Thus, since $\mathcal{I}_{\mathcal{H},\mathcal{A}} \not\models q$, there exists a conjunct $A(t)$ of q with $\pi(t) \notin A^{\mathcal{I}_{\mathcal{H},\mathcal{A}}}$. Since $\mathcal{T}$ and $\mathcal{H}$ are $\mathcal{ELH}_{\mathsf{lhs}}$ TBoxes, $\mathsf{Ind}(\mathcal{A}) = \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} = \Delta^{\mathcal{I}_{\mathcal{H},\mathcal{A}}}$. So we can assume that $\pi(t) = b$, for some $b \in \mathsf{Ind}(\mathcal{A})$. Then $(\mathcal{T}, \mathcal{A}) \models A(b)$ and $(\mathcal{H}, \mathcal{A}) \not\models A(b)$, as required. ❑

Proofs for Non-Polynomial Query Learnability with CQs

We show Lemmas 7 and 8 for the hardness result of learning $\mathcal{EL}_{\mathsf{rhs}}$ TBoxes with CQs. Let q be a CQ. We denote by $\mathsf{Term}(q)$ the set of terms occurring in q. Similarly, the set of atoms occurring in a CQ q is denoted by $\mathsf{Atoms}(q)$. The size $|q|$ of a CQ q is given by the cardinality of $\mathsf{Atoms}(q)$. A CQ q is *connected* if, for all distinct $t, t' \in \mathsf{Terms}(q)$, there exists a sequence $t_0 \cdot r_1 \cdot t_1 \cdot \ldots \cdot t_{k-1} \cdot r_k \cdot t_k$ with $t_0 = t$, $t_k = t'$ and, for all $0 < i \leq k$, there is a role r_i with $r_i(t_{i-1}, t_i) \in \mathsf{Atoms}(q)$.

It is easy to see that if there is a polynomial query learning algorithm with arbitrary CQs, then there is a polynomial query learning algorithm with connected CQs. Thus, from now on we only consider connected CQs.

For a CQ q and an interpretation $\mathcal{I}$, we have $\mathcal{I} \models q$ if, and only if, there is a *match* $\pi : q \to \Delta^\mathcal{I}$ defined as a function from $\mathsf{Terms}(q)$ to $\Delta^\mathcal{I}$, where for $t, t' \in \mathsf{Terms}(q)$:

- if $A(t) \in \mathsf{Atoms}(q)$ then $\pi(t) \in A^\mathcal{I}$;
- if $r(t, t') \in \mathsf{Atoms}(q)$ then $(\pi(t), \pi(t')) \in r^\mathcal{I}$;
- if $t \in \mathsf{N_I}$ then $\pi(t) = t^\mathcal{I} \in \Delta^\mathcal{I}$.

Recall that for any sequence $\boldsymbol{\sigma} = \sigma^1\sigma^2 \ldots \sigma^n$ with $\sigma^i \in \{r, s\}$ and $r, s \in \mathsf{N_R}$, the expression $\exists\boldsymbol{\sigma}.C$ stands for $\exists\sigma^1.\exists\sigma^2 \ldots \exists\sigma^n.C$. For a sequence $\boldsymbol{\sigma}$, let $\mathcal{I}_{\exists\boldsymbol{\sigma}.M}$ be the tree interpretation corresponding to the concept expression $\exists\boldsymbol{\sigma}.M$.

Lemma 7 (restated). 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 $|q|$.

Proof. Assume $(\mathcal{A}, q)$ is given. For each $\boldsymbol{\sigma}$ let $\Pi_{\boldsymbol{\sigma}}$ be the set of all matches $\pi : q \to \mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}$, where $\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}$ is the canonical model of $\mathcal{T}_{\boldsymbol{\sigma}}$ and $\mathcal{A}$. For any $\pi \in \Pi_{\boldsymbol{\sigma}}$, let

$$\mathsf{Im}_{\boldsymbol{\sigma},\pi} = \{p \in \Delta^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}} \mid \pi(t) = p, \text{ for some } t \in \mathsf{Terms}(q)\}.$$

We make the following case distinction.

1. $\mathsf{Im}_\pi \subseteq \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$ for some $\boldsymbol{\sigma}$ and $\pi \in \Pi_{\boldsymbol{\sigma}}$. In this case $(\mathcal{T}_0, \mathcal{A}) \models q$. Thus, for every $\mathcal{T}_{\boldsymbol{\sigma}} \in S$, $(\mathcal{T}_{\boldsymbol{\sigma}}, \mathcal{A}) \models q$, as required.
2. $\mathsf{Im}_\pi \subseteq \Delta^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$ for some $\boldsymbol{\sigma}$ and $\pi \in \Pi_{\boldsymbol{\sigma}}$. In this case, all terms in q are variables (that is, there is no individual from $\mathcal{A}$ occurring in q). There exists $p \in \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$ such that $p \in A^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$ (otherwise $\Delta^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}} = \emptyset$). But then $M^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}} \neq \emptyset$ and so q has a match in the infinite binary tree generated by $M \sqsubseteq \exists r.M \sqcap \exists s.M$. It follows that $(\mathcal{T}_{\boldsymbol{\sigma}}, \mathcal{A}) \models q$ for every $\mathcal{T}_{\boldsymbol{\sigma}} \in S$.

3. For all $\boldsymbol{\sigma}$ and $\pi \in \Pi_{\boldsymbol{\sigma}}$ there are $p_1, p_2 \in \mathsf{Im}_{\boldsymbol{\sigma},\pi}$ such that
   - $p_1 \in \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$;
   - $p_2 \in \Delta^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$.

   We have to show that there are at most $|q|$ distinct $\boldsymbol{\sigma}$ for which $\Pi_{\boldsymbol{\sigma}} \neq \emptyset$. Fix one $\boldsymbol{\sigma}$ with $\Pi_{\boldsymbol{\sigma}} \neq \emptyset$. Assume $\pi \in \Pi_{\boldsymbol{\sigma}}$ is given and let $p_1, p_2 \in \mathsf{Im}_{\boldsymbol{\sigma},\pi}$ such that $p_1 \in \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$ and $p_2 \in \Delta^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$. As $p_2 \in \Delta^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}} \setminus \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$, there is $p_3 \in A^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}}$ such that p_2 is in the tree generated by $A \sqsubseteq \exists \boldsymbol{\sigma}.M$. As q is connected and there is a $p_1 \in \mathsf{Im}_{\boldsymbol{\sigma},\pi} \cap \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$ we have that $p_3 \in \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}} \cap \mathsf{Im}_{\boldsymbol{\sigma},\pi}$. Denote by $\delta_{\boldsymbol{\sigma}}$ the successor of p_3 corresponding to the path $p_3 \cdot \boldsymbol{\sigma} \cdot M$ in $\Delta^{\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}},\mathcal{A}}}$. We have $\delta_{\boldsymbol{\sigma}} \in \mathsf{Im}_{\boldsymbol{\sigma},\pi}$ (since otherwise we find $\pi \in \Pi_{\boldsymbol{\sigma}}$ such that $\mathsf{Im}_{\boldsymbol{\sigma},\pi} \subseteq \Delta^{\mathcal{I}_{\mathcal{T}_0,\mathcal{A}}}$).

   The above holds for any $\boldsymbol{\sigma}$ with $\Pi_{\boldsymbol{\sigma}} \neq \emptyset$. Thus, if the number of $\boldsymbol{\sigma}$ with $\Pi_{\boldsymbol{\sigma}} \neq \emptyset$ exceeds $|q|$, then there is a variable x in q and two distinct $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2$ with matches π_1, π_2 of q in $\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}_1},\mathcal{A}}$ and $\mathcal{I}_{\mathcal{T}_{\boldsymbol{\sigma}_2},\mathcal{A}}$, respectively, such that $\pi_1(x) = \delta_{\boldsymbol{\sigma}_1}$ and $\pi_2(x) = \delta_{\boldsymbol{\sigma}_2}$. This directly leads to a contradiction as one can easily show that the paths $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ have to coincide.

❑

Lemma 8 (restated). 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 $(\mathcal{T}_{\boldsymbol{\sigma}}, \mathcal{A}) \models q$.

Proof. Since every $\mathcal{T}_{\boldsymbol{\sigma}} \in S$ has a concept inclusion $A \sqsubseteq \exists \boldsymbol{\sigma}.M$ (unique for each $\mathcal{T}_{\boldsymbol{\sigma}}$), for every $\mathcal{T}_{\boldsymbol{\sigma}} \in S$, $(\mathcal{T}_{\boldsymbol{\sigma}}, \{A(a)\}) \models \exists x.M(x)$. If $(\mathcal{H}, \{A(a)\}) \not\models \exists x.M(x)$ then the pair $(\{A(a)\}, \exists x.M(x))$ is as required. Otherwise, $(\mathcal{H}, \{A(a)\}) \models \exists x.M(x)$. In this case, there is a sequence of role names $t_1 \cdots t_m$ (which can be empty and, then, we say that $m = 0$) such that $\mathcal{H} \models A \sqsubseteq \exists t_1 \cdots \exists t_m.M$. We write $\exists t_1 \cdots \exists t_m.M(a)$ as an abbreviation for $\exists x_1, \dots, x_m.t_1(a, x_1) \wedge \cdots \wedge t_m(x_{m-1}, x_m) \wedge M(x_m)$. We make the following case distinction:

- If $m < n$ then, since there is no $\mathcal{T}_{\boldsymbol{\sigma}} \in S$ such that $\mathcal{T}_{\boldsymbol{\sigma}} \models A \sqsubseteq \exists t_1 \cdots \exists t_m.M$, we have that there is no $\mathcal{T}_{\boldsymbol{\sigma}} \in S$ such that $(\mathcal{T}_{\boldsymbol{\sigma}}, \{A(a)\}) \models \exists t_1 \cdots \exists t_m.M(a)$. Then the pair $(\{A(a)\}, \exists t_1 \cdots \exists t_m.M(a))$ is as required.
- If $m = n$ then there is exactly one $\mathcal{T}_{\boldsymbol{\sigma}} \in S$ such that $\mathcal{T}_{\boldsymbol{\sigma}} \models A \sqsubseteq \exists t_1 \cdots \exists t_m.M$. Then the pair $(\{A(a)\}, \exists t_1 \cdots \exists t_m.M(a))$ is as required.
- Otherwise $m > n$. In this case let
$$q = \exists x_1, \dots, \exists x_{n+1}.t_1(a, x_1) \wedge \cdots \wedge t_{n+1}(x_n, x_{n+1}),$$
where $t_1, \dots, t_{n+1}$ are the first $n+1$ roles in $\exists t_1 \dots \exists t_m.M$. As $\mathcal{H} \models A \sqsubseteq \exists t_1 \dots \exists t_m.M$, we have that $(\mathcal{H}, \{A(a)\}) \models q$. By definition of $\mathcal{T}_{\boldsymbol{\sigma}}$, there is exactly one $\mathcal{T}_{\boldsymbol{\sigma}} \in S$ such that $(\mathcal{T}_{\boldsymbol{\sigma}}, \{A(a)\}) \models q$, as required.

❑

Proofs for Non-Polynomial Query Learnability of $\mathfrak{F}_{\mathcal{D}}(\mathcal{EL}, \mathcal{EL}$-IQ$)$

We prove that $\mathcal{EL}$ TBoxes are not polynomial query learnable from data retrieval examples with $\mathcal{EL}$-IQs. The proof significantly extends the proof of non-polynomial query learnability of $\mathcal{EL}$ TBoxes from subsumptions (Konev et al. 2014).

We start by giving a brief overview of the construction in (Konev et al. 2014), show that it fails in the data retrieval setting and then demonstrate how it can be modified.

The non-learnability proof in (Konev et al. 2014) proceeds as follows. A learner tries to exactly identify one of the possible target TBoxes $\{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$, for a superpolynomial in n set $\mathfrak{L}_n$ defined below. At every stage of computation the oracle maintains a set of TBoxes S, which the learner is unable to distinguish based on the answers given so far. Initially $S = \{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$. $\{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$ is constructed in such a way that for any $\mathcal{EL}$ inclusion $C \sqsubseteq D$ 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 $|C|$. When a polynomial learner asks a membership query $C \sqsubseteq D$ the oracle answers 'yes' if $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$ and 'no' otherwise. In the latter case the oracle removes polynomially many $\mathcal{T}_L$ such that $\mathcal{T}_L \models C \sqsubseteq D$ from S. Similarly, for any equivalence query with hypothesis $\mathcal{H}$ asked by a polynomial learning algorithm there exists a polynomial size inclusion $C \sqsubseteq D$, which can be returned as a counterexample and that excludes only polynomially many TBoxes from S. Thus, every query to the oracle reduces the size of S at most polynomially in n and the learner cannot distinguish between the remaining TBoxes of the initial superpolynomial set S.

The set of indices $\mathfrak{L}_n$ and the target TBoxes $\mathcal{T}_L$ are defined as follows. Fix two role names r and s. An *n-tuple* L is a sequence of role sequences $(\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, where every $\boldsymbol{\sigma}_i$ is a sequence of role names r and s, that is $\boldsymbol{\sigma}_i = \sigma_i^1 \sigma_i^2 \dots \sigma_i^n$ with $\sigma_i^j \in \{r, s\}$. Then $\mathfrak{L}_n$ is a set of n-tuples such that for 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 $i = j$. There are $N = \lfloor 2^n/n \rfloor$ different tuples in $\mathfrak{L}_n$. For every $n > 0$ and every n-tuple $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$ we define an acyclic $\mathcal{EL}$ TBox $\mathcal{T}_L$ as the union of $\mathcal{T}_0 = \{X_i \sqsubseteq \exists r.X_{i+1} \sqcap \exists s.X_{i+1} \mid 0 \leq i < n\}$ and the following inclusions:

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

where the expression $\exists \boldsymbol{\sigma}.C$ stands for $\exists \sigma^1.\exists \sigma^2 \dots \exists \sigma^n.C$, M is a concept name that 'marks' a designated path given by $\boldsymbol{\sigma}$ and $\mathcal{T}_0$ generates a full binary tree whose edges are labelled with the role names r and s and with X_0 at the root, X_1 at level 1 and so on.

In contrast to the subsumption framework, every $\mathcal{T}_L$ *can* be exactly identified using data retrieval queries. For example, as $X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \cdots \sqcap \exists \boldsymbol{\sigma}_n.M \sqsubseteq A \in \mathcal{T}_L$, a learning from data retrieval queries algorithm can learn all the sequences in the n-tuple $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, by defining an ABox $\mathcal{A} = \{X_0(a_1), r(a_1, a_2), s(a_1, a_2), \dots, r(a_{n-1}, a_n), s(a_{n-1}, a_n), M(a_n)\}$ and then proceed unfolding cycles and minimizing $\mathcal{A}$ via membership queries of the form $(\mathcal{T}_L, \mathcal{A}) \models A(a_1)$.

To show the non-tractability for data retrieval queries, we first modify S in such a way that the concept expression which 'marks' the sequences in $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n)$ is now given by the set $\mathfrak{B}_n$ of all conjunctions $F_1 \sqcap \cdots \sqcap F_n$, where $F_i \in \{E_i, \bar{E}_i\}$, for $1 \leq i \leq n$. Intuitively, every member of $\mathfrak{B}_n$ encodes a binary string of length n with E_i encoding 1 and $\bar{E}_i$ encoding 0. For every $L \in \mathfrak{L}_n$ and every $\mathbf{B} \in \mathfrak{B}_n$ we define $\mathcal{T}_L^{\mathbf{B}}$ as the union of $\mathcal{T}_0$ and the concept inclusions defined above with $\mathbf{B}$ replacing M.

Then for any sequence $\boldsymbol{\sigma}$ of length n there exists at most one $L \in \mathfrak{L}_n$, at most one $1 \leq i \leq n$ and at most one $\mathbf{B} \in \mathfrak{B}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models A_i \sqsubseteq \exists \boldsymbol{\sigma}.\mathbf{B}$ and $\mathcal{T}_L^{\mathbf{B}} \models B_i \sqsubseteq \exists \boldsymbol{\sigma}.\mathbf{B}$. Notice that the size of each $\mathcal{T}_L^{\mathbf{B}}$ is polynomial in n and so $\mathfrak{L}_n$ contains superpolynomially many n-tuples in the size of each $\mathcal{T}_L^{\mathbf{B}}$, with $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$. Every $\mathcal{T}_L^{\mathbf{B}}$ entails, among other inclusions, $\bigsqcap_{i=1}^n C_i \sqsubseteq A$, where every C_i is either A_i or B_i. Let Σ_n be the signature of the TBoxes $\mathcal{T}_L^{\mathbf{B}}$ and consider the TBox $\mathcal{T}^*$ defined as the following set of concept inclusions:

$$\begin{array}{ll} \exists r.(E_1 \sqcap \bar{E}_1) \sqsubseteq (E_1 \sqcap \bar{E}_1) & (E_1 \sqcap \bar{E}_1) \sqsubseteq \exists r.(E_1 \sqcap \bar{E}_1) \\ \exists s.(E_1 \sqcap \bar{E}_1) \sqsubseteq (E_1 \sqcap \bar{E}_1) & (E_1 \sqcap \bar{E}_1) \sqsubseteq \exists s.(E_1 \sqcap \bar{E}_1) \end{array}$$

$$(E_i \sqcap \bar{E}_i) \sqsubseteq A \quad \text{for every } 1 \leq i \leq n \text{ and } A \in \Sigma_n \cap \mathsf{N_C}$$

The basic idea of extending our TBoxes with $\mathcal{T}^*$ is that if $a \in (E_i \sqcap \bar{E}_i)^{\mathcal{I}_\mathcal{A}}$, for an ABox $\mathcal{A}$ and individual $a \in \mathsf{Ind}(\mathcal{A})$, then for all $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$, we have $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models D(b)$, where D is any $\mathcal{EL}$ concept expression over Σ_n and $b \in \mathsf{Ind}(\mathcal{A})$ is any successor or predecessor of a (or a itself). This means that for each individual in $\mathcal{A}$ at most one $\mathbf{B}$ of the 2^n binary strings in $\mathfrak{B}_n$ can be distinguished by data retrieval queries. The following lemma enables us to respond to membership queries without eliminating too many $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$ used to encode $\mathcal{T}_L^{\mathbf{B}}$ in the set of TBoxes that the learner cannot distinguish.

Lemma 54 *For any ABox $\mathcal{A}$, any $\mathcal{EL}$ concept assertion $D(a)$ over Σ_n, and any $a \in \mathsf{Ind}(\mathcal{A})$, if there is $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$ such that $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ then either*

- *$(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$, for every $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$, or*
- *$(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ for at most $|D|$ elements $L \in \mathfrak{L}_n$, or*
- *$(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ for at most $|\mathcal{A}|$ elements $\mathbf{B} \in \mathfrak{B}_n$.*

To show Lemma 54, we first show Lemma 58, which uses Lemmas 56 and 57 from (Konev et al. 2014). We also require the following lemma from (Konev et al. 2012), which characterizes concept inclusions entailed by acyclic $\mathcal{EL}$ TBoxes.

Lemma 55 ((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*

- *if D is a concept name such that $\mathcal{T}$ does not contain an inclusion $D \equiv C$, for some 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 $\mathcal{T} \models C_j \sqsubseteq D'$.*

Lemma 56 *Let $\mathbf{B} = F_1 \sqcap \ldots \sqcap F_n$, where $F_i \in \{E_i, \bar{E}_i\}$. For any $0 \leq m \leq n$, any sequence of role names $\boldsymbol{\sigma} = \sigma^1 \ldots \sigma^m$, any $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n) \in \mathfrak{L}_n$ and any $\mathcal{EL}$ concept expression C over Σ_n, if $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq \exists \boldsymbol{\sigma}.\mathbf{B}$ then either:*

1. *$m = n$, $\boldsymbol{\sigma} = \boldsymbol{\sigma}_i$, for some $1 \leq i \leq n$ 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. *$\models C \sqsubseteq \exists \boldsymbol{\sigma}.\mathbf{B}$.*

Proof. We prove the proposition by induction on m. Since for all F_i occurring in $\mathbf{B}$, $\mathcal{T}_L^{\mathbf{B}}$ does not contain an inclusion $F_i \equiv C$, where C is an $\mathcal{EL}$ concept expression, by Lemma 55, there is a concept name Z such that $\mathcal{T}_L^{\mathbf{B}} \models Z \sqsubseteq F_i$. Then, for $m = 0$, C is of the form $Z \sqcap C'$, where Z is a concept name, C' is an $\mathcal{EL}$ concept expression and $\mathcal{T}_L^{\mathbf{B}} \models Z \sqsubseteq F_i$. This is only possible if Z is F_i itself. As this holds for all F_i, we have that $\models C \sqsubseteq \mathbf{B}$.
For $m > 0$. By Lemma 55 we have one of the following two cases:

- C is of the form $Z \sqcap C'$, for some concept name Z and some $\mathcal{EL}$ concept expression C' such that $\mathcal{T}_L^{\mathbf{B}} \models Z \sqsubseteq \exists \boldsymbol{\sigma}.\mathbf{B}$. It is easy to see that this is only possible if $m = n$, $\boldsymbol{\sigma} = \boldsymbol{\sigma}_i$ and Z is one of A, A_i or B_i.
- C is of the form $\exists \sigma^1.C' \sqcap C''$ for some concept expressions C' and C'' such that $\mathcal{T}_L^{\mathbf{B}} \models C' \sqsubseteq \exists \sigma^2. \cdots \exists \sigma^m.\mathbf{B}$. By induction hypothesis, $\models C' \sqsubseteq \exists \sigma^2. \cdots \exists \sigma^m.\mathbf{B}$. But then $\models C \sqsubseteq \exists \boldsymbol{\sigma}.\mathbf{B}$.

❏

Lemma 57 ((Konev et al. 2014)) *For any acyclic $\mathcal{EL}$ TBox $\mathcal{T}$, any inclusion $A \sqsubseteq C \in \mathcal{T}$ 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 in a position to prove Lemma 58.

Lemma 58 *For all $\mathcal{EL}$ concept inclusions $C \sqsubseteq D$ over Σ_n where $\mathbf{B}$ is not a subconcept of C:*

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

Proof. To prove this lemma we argue by induction on the structure of D and show the following.

Claim 1. For all $\mathcal{EL}$ concept inclusions $C \sqsubseteq D$ over Σ_n where $\mathbf{B} \in \mathfrak{B}_n$ is not a subconcept of C, if there is $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D$ then:

- either $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$ and every $\mathbf{B} \in \mathfrak{B}_n$ or
- for each $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D$ there is $\boldsymbol{\sigma}$ in L and a sequence of roles $t_1, \ldots, t_m$, $m \geq 0$, such that $\models D \sqsubseteq \exists t_1. \cdots \exists t_m. \exists \boldsymbol{\sigma}.\top$, where $t_j \in \{r, s\}$, $1 \leq j \leq m$.

We assume throughout the proof that in all cases $\mathbf{B}$ is not a subconcept of C and that there exists some $L_0 \in \mathfrak{L}_n$ such that $\mathcal{T}_{L_0}^{\mathbf{B}} \models C \sqsubseteq D$.

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

- D is one of X_i, A_i, B_i, E_i or $\bar{E}_i$ for $1 \leq i \leq n$. By Lemma 55, C is of the form $Z \sqcap C'$, for some concept name Z, and $\mathcal{T}_{L_0}^{\mathbf{B}} \models Z \sqsubseteq D$. If D is one of X_i, A_i, B_i, E_i or $\bar{E}_i$, then this can only be the case if $Z = D$. But then for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D$.
- D is X_0. By Lemma 55, C is of the form $Z \sqcap C'$, for some concept name Z, and $\mathcal{T}_{L_0}^{\mathbf{B}} \models Z \sqsubseteq X_0$. This is the case if either $Z = X_0$, or Z is one of A, A_i, B_i, $1 \leq i \leq n$. In either case, for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq X_0$.
- D is A. If C is of the form $A \sqcap C'$ or, for all i, $1 \leq i \leq n$, A_i or B_i is a conjunct of C, then for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq A$. 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 $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq A$. Notice that $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq A$, for $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n) \in \mathfrak{L}_n$, if, and only if, $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq X_0 \sqcap \exists \boldsymbol{\sigma}_1.\mathbf{B} \sqcap \cdots \sqcap \exists \boldsymbol{\sigma}_n.\mathbf{B}$. By Lemma 56, for such a $\mathcal{T}_L^{\mathbf{B}}$ we must have $\models C \sqsubseteq \exists \boldsymbol{\sigma}_j.\mathbf{B}$, but then this is not possible as $\mathbf{B}$ is not a subconcept of C.

Thus if D is a concept name then either for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D$ or there exists no $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D$, where $\mathbf{B}$ is not a subconcept of C.

Induction step. If $D = D_1 \sqcap D_2$, then $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D$ if, and only if, $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D_i$, $i \in \{1, 2\}$. So the lemma follows from the induction hypothesis.

For $D = \exists t.D'$, suppose that there is $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq D$. Then, by Lemma 55, either (i) there exists a conjunct Z of C, Z a concept name, such that $\mathcal{T}_L^{\mathbf{B}} \models Z \sqsubseteq \exists t.D'$ or (ii) there exists a conjunct $\exists t.C'$ of C with $\mathcal{T}_L^{\mathbf{B}} \models C' \sqsubseteq D'$. Consider cases (i) and (ii).

(i) Let Z be a conjunct of C such that Z is a concept name and $\mathcal{T}_L^{\mathbf{B}} \models Z \sqsubseteq \exists t.D'$. Notice that Z cannot be E_i or $\bar{E}_i$ as for no $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models E_i \sqsubseteq \exists t.D'$ or $\mathcal{T}_L^{\mathbf{B}} \models \bar{E}_i \sqsubseteq \exists t.D'$. Consider the remaining possibilities.

- Z is one of X_i, $0 \leq i \leq n$. It is easy to see that for $L, L' \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models X_i \sqsubseteq \exists t.D'$ if, and only if $\mathcal{T}_{L'}^{\mathbf{B}} \models X_i \sqsubseteq \exists t.D'$. Thus, for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models Z \sqsubseteq \exists t.D'$.
- Z is one of A_i, B_i for $1 \leq i \leq n$. By Lemma 57, $\mathcal{T}_L^{\mathbf{B}} \models Z \sqsubseteq \exists t.D'$ if, and only if, $\mathcal{T}_L^{\mathbf{B}} \models X_0 \sqcap \exists \boldsymbol{\sigma}_i.\mathbf{B} \sqsubseteq \exists t.D'$. By Lemma 55, either $\mathcal{T}_L^{\mathbf{B}} \models X_0 \sqsubseteq \exists t.D'$ or $\mathcal{T}_L^{\mathbf{B}} \models \exists \boldsymbol{\sigma}_i.\mathbf{B} \sqsubseteq \exists t.D'$. If $\mathcal{T}_L^{\mathbf{B}} \models X_0 \sqsubseteq \exists t.D'$ then for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq \exists t.D'$. Now, suppose that $\exists t.D'$ is such that $\mathcal{T}_L^{\mathbf{B}} \not\models X_0 \sqsubseteq \exists t.D'$ and $\mathcal{T}_L^{\mathbf{B}} \models \exists \boldsymbol{\sigma}_i.\mathbf{B} \sqsubseteq \exists t.D'$. By inductive applications of Lemma 55, this is only possible when $\models \exists t.D' \sqsubseteq \exists \boldsymbol{\sigma}_i.\top$. Notice that since all $\boldsymbol{\sigma}_i$ are unique, there exists exactly one $L \in \mathfrak{L}_n$ (namely, L is L_0) such that $\mathcal{T}_L^{\mathbf{B}} \models Z \sqsubseteq \exists \boldsymbol{\sigma}_i.F$, where $\models \mathbf{B} \sqsubseteq F$.
- Z is A. Suppose that for some $L = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n) \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models A \sqsubseteq \exists t.D'$, equivalently $\mathcal{T}_L^{\mathbf{B}} \models X_0 \sqcap \exists \boldsymbol{\sigma}_1.\mathbf{B} \sqcap \ldots \exists \boldsymbol{\sigma}_n.\mathbf{B} \sqsubseteq \exists t.D'$. By Lemma 55, either $\mathcal{T}_L^{\mathbf{B}} \models X_0 \sqsubseteq \exists t.D'$ or $\mathcal{T}_L^{\mathbf{B}} \models \exists \boldsymbol{\sigma}_i.\mathbf{B} \sqsubseteq \exists t.D'$, for some $i : 1 \leq i \leq n$, so, as above, unless $\mathcal{T}_L^{\mathbf{B}} \models X_0 \sqsubseteq \exists t.D'$ we have that $\models \exists t.D' \sqsubseteq \exists \boldsymbol{\sigma}_i.\top$, as required.

(ii) Let $\exists t.C'$ be a conjunct of C with $\mathcal{T}_L^{\mathbf{B}} \models C' \sqsubseteq D'$. The induction hypothesis implies that either (a) for every $L \in \mathfrak{L}_n$ we have that $\mathcal{T}_L^{\mathbf{B}} \models C' \sqsubseteq D'$ or (b) for each $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models C' \sqsubseteq D'$ there is $\boldsymbol{\sigma}$ in L and a sequence of roles $t_1, \ldots, t_m$, $m \geq 0$, such that $\models D' \sqsubseteq \exists t_1. \cdots \exists t_m.\exists \boldsymbol{\sigma}.\top$, where $t_j \in \{r, s\}$, $1 \leq j \leq m$. In case (a), we have that for every $L \in \mathfrak{L}_n$, $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq \exists t.D'$. In case (b), if for each $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models C' \sqsubseteq D'$ there is $\boldsymbol{\sigma}$ such that $\models D' \sqsubseteq \exists t_1. \ldots \exists t_m.\exists \boldsymbol{\sigma}.\top$ then same happens with $\exists t.D'$ (notice that for every $L \in \mathfrak{L}_n$ and every $\mathbf{B} \in \mathfrak{B}_n$ we have that $\mathcal{T}_L^{\mathbf{B}} \models C' \sqsubseteq D'$ iff $\mathcal{T}_L^{\mathbf{B}} \models \exists t.C' \sqsubseteq \exists t.D'$).

To summarize, either $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq \exists t.D'$ for every $L \in \mathfrak{L}_n$ and every $\mathbf{B} \in \mathfrak{B}_n$ or $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq \exists t.D'$ implies that $\models \exists t.D' \sqsubseteq \exists t_0. \ldots \exists t_m.\exists \boldsymbol{\sigma}.\top$, $m \geq 0$, for some $\boldsymbol{\sigma}$ in L. Since all $\boldsymbol{\sigma}$ are unique for each $L \in \mathfrak{L}_n$, the number of different $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models C \sqsubseteq \exists t.D'$ does not exceed $|D|$. ❑

Before we proceed to the proof of Lemma 54, we need Lemma 60.

Definition 59 *The unravelling $\mathcal{A}^u$ of $\mathcal{A}$ into a (possibly infinite) tree is defined as:*

- $\mathsf{Ind}(\mathcal{A}^u)$ *is the set of sequences* $b_0 r_0 \cdots r_{n-1} b_n$ *with* $b_0, \ldots, b_n \in \mathsf{Ind}(\mathcal{A})$, $r_0, \ldots, r_{n-1} \in \mathsf{N_R}$ *and* $r_i(b_i, b_{i+1}) \in \mathcal{A}$*;*
- *for each* $A(b) \in \mathcal{A}$ *and* $\alpha = b_0 r_0 \cdots r_{n-1} \cdot b_n \in \mathsf{Ind}(\mathcal{A}^u)$ *with* $b_n = b$*, we have* $A(\alpha) \in \mathcal{A}^u$*;*
- *for each* $\alpha = b_0 r_0 \cdots r_{n-1} b_n \in \mathsf{Ind}(\mathcal{A}^u)$ *with* $n > 0$*, we have* $r_{n-1}(b_0 r_0 \cdots r_{n-2} b_{n-1}, \alpha) \in \mathcal{A}^u$.

Lemma 60 *For any ABox $\mathcal{A}$ and $\mathcal{EL}$ concept expression D over Σ_n there is a concept expression $C_\mathcal{A}$ such that $a \in C_\mathcal{A}^{\mathcal{I}_\mathcal{A}}$ and, for every $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$:*

$$(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models D(a) \quad \textit{iff} \quad \mathcal{T}_L^{\mathbf{B}} \models C_\mathcal{A} \sqsubseteq D.$$

Proof. Let $\mathcal{A}^u$ be the unravelling of $\mathcal{A}$. Let $\mathcal{T}_L^{\mathbf{B}}$ be a TBox for some arbitrary $\mathbf{B} \in \mathfrak{B}_n$ and $L \in \mathfrak{L}_n$. By definition of $\mathcal{A}^u$ we have that $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models D(a)$ iff $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}^u) \models D(a)$. Denote as $\mathcal{A}_a^{u,k}$ the subtree of $\mathcal{A}^u$ which is rooted in $a \in \mathsf{Ind}(\mathcal{A}^u)$ and has depth $k \in \mathbb{N}$. Let $\mathcal{T}_L^{\mathbf{B}'}$ be the result of removing $X_0 \sqcap \exists \boldsymbol{\sigma}_1.\mathbf{B} \sqcap \cdots \sqcap \exists \boldsymbol{\sigma}_n.\mathbf{B} \sqsubseteq A$ from $\mathcal{T}_L^{\mathbf{B}}$. Then, $(\mathcal{T}_L^{\mathbf{B}'}, \mathcal{A}^u) \models D(a)$ iff $(\mathcal{T}_L^{\mathbf{B}'}, \mathcal{A}_a^{u,|D|}) \models D(a)$. Let $\mathcal{I}_{\mathcal{T}_L^{\mathbf{B}'}, \mathcal{A}^u}$ be the canonical model of $\mathcal{T}_L^{\mathbf{B}'}$ and $\mathcal{A}^u$. By definition of $\mathcal{T}_L^{\mathbf{B}}$, one can make it a canonical model of $\mathcal{T}_L^{\mathbf{B}}$ and $\mathcal{A}^u$ by including $d \in A^{\mathcal{I}_{\mathcal{T}_L^{\mathbf{B}'}, \mathcal{A}^u}}$ whenever $d \in (X_0 \sqcap \exists \boldsymbol{\sigma}_1.\mathbf{B} \sqcap \cdots \sqcap \exists \boldsymbol{\sigma}_n.\mathbf{B})^{\mathcal{I}_{\mathcal{T}_L^{\mathbf{B}'}, \mathcal{A}^u}}$. Then, $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}^u) \models D(a)$ iff $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}_a^{u,|D|+n}) \models D(a)$. Let $C_\mathcal{A}$ be the concept expression corresponding to the tree interpretation of $\mathcal{A}_a^{u,|D|+n}$ rooted in a. We have that, for every $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$, $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models D(a)$ iff $\mathcal{T}_L^{\mathbf{B}} \models C_{\mathcal{A}} \sqsubseteq D$. ❑

We can now proceed to the proof of Lemma 54. We say that an $\mathcal{EL}$ concept expression C *occurs* in an ABox $\mathcal{A}$ if there exists $a \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A} \models C(a)$. For $a, b \in \mathsf{Ind}(\mathcal{A})$, a *role chain* from a to b is a sequence $a_0 \cdot t_0 \cdot \ldots \cdot t_{n-1} \cdot a_n$ with $a_0 = a$, $a_n = b$ and $t_i(a_i, a_{i+1}) \in \mathcal{A}$, where $0 \leq i \leq n-1$ and $t_i \in \{r, s\}$.

Lemma 54 (restated). For any ABox $\mathcal{A}$, any $\mathcal{EL}$ concept assertion $D(a)$ over Σ_n, and any $a \in \mathsf{Ind}(\mathcal{A})$, if there is $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$ such that $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ then:

- either $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$, for every $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$, or
- $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ for at most $|D|$ elements $L \in \mathfrak{L}_n$, or
- $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ for at most $|\mathcal{A}|$ elements $\mathbf{B} \in \mathfrak{B}_n$.

Proof. We make a case distinction:

1. for all i, $1 \leq i \leq n$, $E_i \sqcap \bar{E}_i$ does not occur in $\mathcal{A}$: first notice that in this case, for every $\mathcal{EL}$ concept expression C over Σ_n, $a \in \mathsf{Ind}(\mathcal{A})$ and $\mathcal{T}_L^{\mathbf{B}} \in S$:
$$(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models C(a) \quad \text{iff} \quad (\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models C(a).$$
For any $\mathcal{A}$ and $\mathcal{EL}$ concept expression D over Σ_n, by Lemma 60, there is a concept expression $C_{\mathcal{A}}$ such that $a \in C_{\mathcal{A}}^{\mathcal{I}_{\mathcal{A}}}$ and, for every $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$:
$$(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models D(a) \quad \text{iff} \quad \mathcal{T}_L^{\mathbf{B}} \models C_{\mathcal{A}} \sqsubseteq D.$$
If there is no $\mathbf{B} \in \mathfrak{B}_n$ such that $\mathbf{B}$ occurs in $\mathcal{A}$ then the Lemma follows from Corollary 58. Notice that although our construction of $C_{\mathcal{A}}$ is not polynomial, Corollary 58 does not impose any restriction in the size of $C_{\mathcal{A}}$. Otherwise, since for all i, $1 \leq i \leq n$, $E_i \sqcap \bar{E}_i$ does not occur in $\mathcal{A}$, we have that the number of $\mathbf{B} \in \mathfrak{B}_n$ such that $\mathbf{B}$ occurs in $\mathcal{A}$ is linear $|\mathcal{A}|$. So the number of $\mathbf{B} \in \mathfrak{B}_n$ such that $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ does not exceed $|\mathcal{A}|$.
2. there is i, $1 \leq i \leq n$, such that $E_i \sqcap \bar{E}_i$ occurs in $\mathcal{A}$: let $E_i \sqcap \bar{E}_{i\mathcal{A}}$ be the set of individuals $b \in \mathsf{Ind}(\mathcal{A})$ such that $E_i \sqcap \bar{E}_i(b) \in \mathcal{A}$. By construction of $\mathcal{T}^*$, for every ABox $\mathcal{A}$ and every $\mathcal{EL}$ concept expression D over Σ_n we have that $(\mathcal{T}^*, \mathcal{A}) \models D(b)$, where $b \in E_i \sqcap \bar{E}_{i\mathcal{A}}$. Then, in particular, for every $L \in \mathfrak{L}_n$ we have that $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(b)$. For $a \in \mathsf{Ind}(\mathcal{A}) \setminus E_i \sqcap \bar{E}_{i\mathcal{A}}$ we make a case distinction:
    - there is a role chain from a to some $b \in E_i \sqcap \bar{E}_{i\mathcal{A}}$: by definition of $\mathcal{T}^*$, as $(E_i \sqcap \bar{E}_i) \sqsubseteq A$ for every $1 \leq i \leq n$ and every $A \in \Sigma_n \cap \mathsf{N_C}$, we have that $(\mathcal{T}^*, \mathcal{A}) \models (E_1 \sqcap \bar{E}_1)(b)$. Then, since $\{\exists r.(E_1 \sqcap \bar{E}_1) \sqsubseteq (E_1 \sqcap \bar{E}_1), \exists s.(E_1 \sqcap \bar{E}_1) \sqsubseteq (E_1 \sqcap \bar{E}_1)\} \subseteq \mathcal{T}^*$, we have that $(\mathcal{T}^*, \mathcal{A}) \models (E_1 \sqcap \bar{E}_1)(a)$. In this case, by the argument above, for every $L \in \mathfrak{L}_n$ and every $\mathcal{EL}$ concept expression D over Σ_n, we have that $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$.
    - for all $b \in E_i \sqcap \bar{E}_{i\mathcal{A}}$, there is no role chain from a to b: let $\mathcal{A}' = \mathcal{A} \setminus \{E_i(b), \bar{E}_i(b) \mid b \in E_i \sqcap \bar{E}_{i\mathcal{A}}\}$. Since in this case, for all $b \in E_i \sqcap \bar{E}_{i\mathcal{A}}$, there is no role chain from a to b, we have that, for every $\mathcal{EL}$ concept expression D, $\mathcal{A} \models D(a)$ iff $\mathcal{A}' \models D(a)$. By definition of $\mathcal{A}'$, $E_i \sqcap \bar{E}_i$ does not occur in $\mathcal{A}'$, then the lemma follows as in Case 1.

❑

The next lemma from (Konev et al. 2014) prepares the proof of Lemma 62.

Lemma 61 ((Konev et al. 2014)) *For any $0 \leq i \leq n$ and Σ_n-concept D, if $\mathcal{T}_0 \not\models X_i \sqsubseteq D$ then there exists a sequence of role names $t_1, \ldots t_l$ such that $\models D \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ 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, $0 \leq l \leq n - i + 1$.*

Lemma 62 is immediate from Lemma 15 presented in (Konev et al. 2014). It shows how the oracle can answer equivalence queries eliminating at most one $L \in \mathfrak{L}_n$ used to encode $\mathcal{T}_L^{\mathbf{B}}$ in the set S of TBoxes that the learner cannot distinguish.

Lemma 62 *For any $n > 1$ and any $\mathcal{EL}$ TBox $\mathcal{H}$ in Σ_n with $|\mathcal{H}| < 2^n$, there exists an ABox $\mathcal{A}$, an individual $a \in \mathsf{Ind}(\mathcal{A})$ and an $\mathcal{EL}$ concept expression D over Σ_n such that (i) $|\mathcal{A}| + |D|$ does not exceed $6n$ and (ii) if $(\mathcal{H}, \mathcal{A}) \models D(a)$ then $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models D(a)$ for at most one $L \in \mathfrak{L}_n$ and if $(\mathcal{H}, \mathcal{A}) \not\models D(a)$ then for every $L \in \mathfrak{L}_n$ we have $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$.*

Proof. As $\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^* \models C \sqsubseteq D$ iff $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}_C) \models D(\rho_C)$, where $\mathcal{A}_C$ is a tree shaped ABox (rooted in $\rho_C \in \mathsf{Ind}(\mathcal{A}_C)$) corresponding to the $\mathcal{EL}$ concept expression C, to prove this lemma we show the following claim.

Claim 1. 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 (i) the size of $C \sqsubseteq D$ does not exceed $6n$ and (ii) if $\mathcal{H} \models C \sqsubseteq D$ then $\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^* \models C \sqsubseteq D$ for at most one $L \in \mathfrak{L}_n$ and if $\mathcal{H} \not\models C \sqsubseteq D$ then for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^* \models C \sqsubseteq D$.

We define an exponentially large TBox $\mathcal{T}_\sqcap$ and use it to prove that one can select an $\mathcal{EL}$ concept inclusion $C \sqsubseteq D$ in such a way that either $\mathcal{H} \models C \sqsubseteq D$ and $\mathcal{T}_\sqcap \not\models C \sqsubseteq D$, or vice versa. Then, the oracle can return $(\mathcal{A}_C, D(\rho_C))$ as a counterexample, where $\mathcal{A}_C$ is a tree shaped ABox (rooted in $\rho_C \in \mathsf{Ind}(\mathcal{A}_C)$) corresponding to the $\mathcal{EL}$ concept expression C.

To define $\mathcal{T}_\sqcap$, for any sequence $\boldsymbol{b} = b_1 \ldots b_n$, where every b_i is either 0 or 1, we denote 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
$$\mathcal{T}_\sqcap = \mathcal{T}_0 \cup \{C_{\boldsymbol{b}} \sqsubseteq A \sqcap X_0 \mid \boldsymbol{b} \in \{0,1\}^n\}.$$
Let $\mathcal{A}_{C_{\boldsymbol{b}}}$ be the ABox corresponding to a concept expression $C_{\boldsymbol{b}}$, as defined above. Since, for all i, $1 \leq i \leq n$, $E_i \sqcap \bar{E}_i$ does not occur in $\mathcal{A}_{C_{\boldsymbol{b}}}$, we have that $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}_{C_{\boldsymbol{b}}}) \models C(a)$ iff $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}_{C_{\boldsymbol{b}}}) \models C(a)$. Then, in the following we only consider $\mathcal{T}_L^{\mathbf{B}}$. Consider the possibilities for $\mathcal{H}$ and $\mathcal{T}_\sqcap$.

(1) If $\mathcal{H} \not\models \mathcal{T}_\cap$ then there exists an inclusion $C \sqsubseteq D \in \mathcal{T}_\cap$ such that $\mathcal{H} \not\models C \sqsubseteq D$. Clearly, $C \sqsubseteq D$ is entailed by $\mathcal{T}_L^{\mathbf{B}}$, for every $L \in \mathfrak{L}_n$, and the size of $C \sqsubseteq D$ does not exceed $6n$, so $C \sqsubseteq D$ is as required.

(2) Suppose that for some $\boldsymbol{b} \in \{0,1\}^n$ and a concept expression of the form $\exists t.D'$ we have $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq \exists t.D'$ and $\mathcal{T}_\cap \not\models C_{\boldsymbol{b}} \sqsubseteq \exists t.D'$. To 'minimise' $C_{\boldsymbol{b}} \sqsubseteq \exists t.D'$, notice that $\mathcal{T}_0 \not\models X_0 \sqsubseteq \exists t.D'$. Then, by Lemma 61, there exists a sequence of role names $t_1, \ldots, t_l$, for $0 \leq l \leq n+1$ and Y being $\top$ or a concept name such that $\models \exists t.D' \sqsubseteq \exists t_1. \cdots \exists t_l.Y$, so $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq \exists t_1. \cdots \exists t_l.Y$, and $\mathcal{T}_0 \not\models X_0 \sqsubseteq \exists t_1. \cdots \exists t_l.Y$. 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 $\mathcal{T}_L^{\mathbf{B}} \models C_{\boldsymbol{b}} \sqsubseteq \exists t_1 \cdots \exists t_l.Y$ for at most one $L \in \mathfrak{L}_n$.

Suppose for some $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models C_{\boldsymbol{b}} \sqsubseteq \exists t_1. \cdots \exists t_l.Y$. By Lemma 55, there is A_j or B_j such that $\mathcal{T}_L^{\mathbf{B}} \models A_j \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ (or $\mathcal{T}_L^{\mathbf{B}} \models B_j \sqsubseteq \exists t_1. \cdots \exists t_l.Y$, respectively). As $\mathcal{T}_0 \not\models X_0 \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ it is easy to see that this is only possible when $l = n$, $(t_1, t_2, \ldots, t_n) = \boldsymbol{\sigma}_j$, and Y is implied by $\mathbf{B}$. Since every $\boldsymbol{\sigma}_j$ is unique, for every $L' \in \mathfrak{L}_n$ such that $L' \neq L$ we have $\mathcal{T}_{L'}^{\mathbf{B}} \not\models C_{\boldsymbol{b}} \sqsubseteq \exists \boldsymbol{\sigma}_j.Y$.

Thus, $C_{\boldsymbol{b}} \sqsubseteq \exists t_1. \cdots \exists t_l.Y$ is as required.

(3) Finally, suppose that Case 1 and 2 above do not 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 an inclusion $C \sqsubseteq D$ satisfying the conditions of the lemma, $\mathcal{H}$ contains at least 2^n different inclusions. Thus, we have derived a contradiction.

Fix $\boldsymbol{b} \in \{0,1\}^n$. As $\mathcal{H} \models \mathcal{T}_\cap$ we have $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq A$. Then there must exist an (at least one) inclusion $C \sqsubseteq A \sqcap D \in \mathcal{H}$ such that $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq C$ and $\not\models C \sqsubseteq A$. Let $C = Z_1 \sqcap \cdots \sqcap Z_m \sqcap \exists t_1.C'_1 \sqcap \cdots \sqcap \exists t_l.C'_l$, where $Z_1, \ldots, Z_m$ 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 $j = 1, \ldots l$. As $\mathcal{H} \models \mathcal{T}_\cap$ we have $\mathcal{H} \models X_0 \sqsubseteq \exists t_j.C'_j$, for $j = 1, \ldots l$. So $\mathcal{H} \models Z_1 \sqcap \cdots \sqcap Z_m \sqcap X_0 \sqsubseteq A$.

Suppose that for some $i : 1 \leq i \leq n$ there exists no $j : 1 \leq j \leq m$ such that Z_j is either A_i or B_i. Then we have $\mathcal{T}_L^{\mathbf{B}} \not\models Z_1 \sqcap \cdots \sqcap Z_m \sqcap X_0 \sqsubseteq A$, for any $L \in \mathfrak{L}_n$. Notice that in the worst case $Z_1 \sqcap \cdots \sqcap Z_m$ contains the conjunction of all Σ_n-concept names, except A_i, B_i, so the size of $Z_1 \sqcap \cdots \sqcap Z_m \sqcap X_0 \sqsubseteq A$ does not exceed $6n$, and $Z_1 \sqcap \cdots \sqcap Z_m \sqcap X_0 \sqsubseteq A$ is as required.

Assume that $Z_0 \sqcap \cdots \sqcap Z_m \sqcap X_0$ contains a conjunct B_i such that $b_i \neq 0$. Then $\mathcal{H} \models C_{\boldsymbol{b}} \sqsubseteq B_i$ and for no $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models C_{\boldsymbol{b}} \sqsubseteq B_i$. The size of $C_{\boldsymbol{b}} \sqsubseteq B_i$ does not exceed $6n$, so it 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 for no $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L^{\mathbf{B}} \models C_{\boldsymbol{b}} \sqsubseteq A_i$. The size of $C_{\boldsymbol{b}} \sqsubseteq A_i$ does not exceed $6n$, so it is as required.

The only remaining option is that $Z_1 \sqcap \cdots \sqcap Z_m \sqcap X_0$ 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 inclusion $C \sqsubseteq D$ satisfying the conditions of the lemma then $\mathcal{H}$ contains at least 2^n inclusions. ❑

Intuitively, by Lemmas 54 and 62, we have that: (i) any polynomial size membership query can distinguish at most polynomially many TBoxes from $S = \{\mathcal{T}_L^{\mathbf{B}} \mid L \in \mathfrak{L}_n \text{ and } \mathbf{B} \in \mathfrak{B}_n\}$; and (ii) if the learner's hypothesis is polynomial size then there exists a polynomial size counterexample that the oracle can give which distinguishes at most polynomially many TBoxes from S. We can now state our theorem and give a formal argument.

Theorem 63 *The learning framework $\mathfrak{F}_\mathcal{D}(\mathcal{EL}, \mathcal{EL}$-IQ) is not polynomial query exact learnable.*

Proof. Assume that TBoxes are polynomial query learnable in the data retrieval setting. Then there exists a learning algorithm whose sum of inputs to queries is bounded at any stage by a polynomial $p(n,m)$, where $k \cdot n$, $k \in \mathbb{N}$, bounds the size of the target and m is the largest counterexample seen so far. Choose n such that $\lfloor 2^n/n \rfloor > (p(n,6n))^2$ and let $S = \{\mathcal{T}_L^{\mathbf{B}} \mid L \in \mathfrak{L}_n \text{ and } \mathbf{B} \in \mathfrak{B}_n\}$. We follow Angluin's strategy of removing elements from S in such a way that the learner cannot distinguish between any of the remaining $\mathcal{T}_L^{\mathbf{B}}$ TBoxes encoded by $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$. The strategy is as follows.

Given an membership query with the data retrieval example $(\mathcal{A}, D(a))$ as input, if $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ for every $L \in \mathfrak{L}_n$ and every $\mathbf{B} \in \mathfrak{B}_n$, then the answer is 'yes'; otherwise the answer is 'no' and all $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$ with $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ are removed from S. By Lemma 54, at most $|\mathcal{A}| + |D|$ elements can be removed from S. Given an equivalence query with $\mathcal{H}$, the answer is 'no' and a positive data retrieval counterexample $(\mathcal{A}, D(a))$ with size bounded by $6n$ is guaranteed by Lemma 62.

As all counterexamples produced are bounded by $6n$, the sum of the sizes of inputs to queries made by the learning algorithm is bounded by $p(n,6n)$. Hence, by Lemmas 54 and 62, at most $p(n,6n)$ elements are removed from S during the run of the algorithm. But then the algorithm cannot distinguish between any of the remaining TBoxes $\mathcal{T}_L^{\mathbf{B}}$ and $\mathcal{T}_{L'}^{\mathbf{B}'}$ for $L \neq L'$ or $\mathbf{B} \neq \mathbf{B}'$ based on the given answers and, thus, we have derived a contradiction. ❑

We note that the non polynomial query learnability proof for $\mathfrak{F}_\mathcal{D}(\mathcal{EL}, \mathcal{EL}$-IQ) given above can be easily extended to $\mathfrak{F}_\mathcal{D}(\mathcal{ELH}, \mathcal{EL}$-IQ): the construction above does not use RIs and in any equivalence query with an RI $r \sqsubseteq s$ in the hypothesis (where $\mathcal{T} \not\models r \not\equiv s$), the oracle can return $(\{r(a,b)\}, s(a,b))$ as a negative counterexample.