Exact Learning Description Logic Ontologies from Data Retrieval Examples

Boris Konev, Ana Ozaki, and Frank Wolter

Department of Computer Science, University of Liverpool, UK

Abstract. We investigate the complexity of learning description logic ontologies in Angluin et al.'s framework of exact learning via queries posed to an oracle. We consider membership queries of the form "is individual a a certain answer to a data retrieval query q in a given ABox and the unkown target TBox?" and equivalence queries of the form "is a given TBox equivalent to the unknown target TBox?". We show that (i) DL-Lite TBoxes with role inclusions and $\mathcal{ELI}$ concept expressions on the right-hand side of inclusions and (ii) $\mathcal{EL}$ TBoxes without complex concept expressions on the right-hand side of inclusions can be learned in polynomial time. Both results are proved by a non-trivial reduction to learning from subsumption examples. We also show that arbitrary $\mathcal{EL}$ TBoxes cannot be learned in polynomial time.

1 Introduction

Building an ontology is prone to errors, time consuming, and costly. The research communities has addressed this problem in many different ways, for example, by supplying tool support for editing ontologies [15, 4, 9], developing reasoning support for debugging ontologies [18], supporting modular ontology design [17], and by investigating automated ontology generation from data or text [8, 6, 5, 14]. 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 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 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 instance retrieval examples:

- assume a data instance $\mathcal{A}$ and a query $q(x)$ are given. Is the individual a a certain answer to query $q(x)$ 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 $q(x)$, and an individual a such that a is a certain answer to $q(x)$ in $\mathcal{A}$ and the ontology $\mathcal{O}$ and it is not a certain answer to $q(x)$ 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 ontologies $\mathcal{H}$ 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 [8]. 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 [1]. The queries using instance 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 learns an arbitrary ontology over a given ontology language in polynomial time. We consider polynomial learnability in three distinct DLs: we show that DL-Lite ontologies with role inclusions and arbitrary $\mathcal{ELI}$ concepts on the right-hand side of concept inclusions can be learned in polynomial time if database queries in instance retrieval examples are $\mathcal{ELI}$ instance queries (or, equivalently, acyclic conjunctive queries). We call this DL DL-Lite$_{\mathcal{R}}^{\exists}$ and note that it is the core of the web ontology language profile OWL2 QL. We also note that *without* complex $\mathcal{ELI}$ concepts on the right-hand side of concept inclusions, polynomial learnability would be trivial as only finitely many non-equivalent such TBoxes exist over a given vocabulary of concept and role names. The second DL we consider is $\mathcal{EL}$ which is the logic underpinning the web ontology language profile OWL2 EL. We show that $\mathcal{EL}$ TBoxes cannot be learned in polynomial time using the protocol above if the database queries in instance retrieval examples are $\mathcal{EL}$ instance queries. We then consider the fragment $\mathcal{EL}_{\mathsf{lhs}}$ of $\mathcal{EL}$ without complex concepts on the right-hand side of concept inclusions and prove that it can be learned in polynomial time using the above protocol with instance retrieval examples. The proofs of the positive learning results are by reduction to polynomial time learnability results for DL-Lite$_{\mathcal{R}}^{\exists}$ and $\mathcal{EL}_{\mathsf{lhs}}$ for the case in which *concept subsumptions* rather than instance retrieval examples are used in the communication between the learner and the

oracle [12]. Our move from concept subsumptions to data retrieval examples 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. Detailed proofs are provided at `http://csc.liv.ac.uk/~frank/publ/publ.html`.

2 Preliminaries

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite sets of *concept* and *role* names, respectively. The dialect DL-Lite$_{\mathcal{R}}^{\exists}$ of DL-Lite is defined as follows [7]. A *role* is a role name or an inverse role r^- with $r \in \mathsf{N_R}$. A *role inclusion (RI)* is of the form $r \sqsubseteq s$, where r and s are roles. A *basic concept* is either a concept name or of the form $\exists r.\top$, with r a role. A DL-Lite$_{\mathcal{R}}^{\exists}$ concept inclusion (CI) is of the form $B \sqsubseteq C$, where B is a basic concept expression and C is an $\mathcal{ELI}$ concept expression, that is, C is formed according to the rule $C, D := A \mid \top \mid C \sqcap D \mid \exists r.C \mid \exists r^-.C$ where A ranges over $\mathsf{N_C}$ and r ranges over $\mathsf{N_R}$. A DL-Lite$_{\mathcal{R}}^{\exists}$ TBox is a finite set of DL-Lite$_{\mathcal{R}}^{\exists}$ CIs and RIs. As usual, an $\mathcal{EL}$ *concept expression* is an $\mathcal{ELI}$ concept expression that does not use inverse roles, an $\mathcal{EL}$ *concept inclusion* has the form $C \sqsubseteq D$ with C and D $\mathcal{EL}$ concept expressions, and a *(general) $\mathcal{EL}$ TBox* is a finite set of $\mathcal{EL}$ concept inclusions [2]. We also consider the restriction $\mathcal{EL}_{\mathsf{lhs}}$ of general $\mathcal{EL}$ TBoxes where only concept names are allowed on the right-hand side of concept inclusions. The *size* of a concept expression C, denoted with $|C|$, is the length of the string that represents it, where concept names and role names are considered to be of length one. A TBox *signature* is the set of concept and role names occurring in the TBox. The *size* of a TBox $\mathcal{T}$, denoted with $|\mathcal{T}|$, is $\sum_{C \sqsubseteq D \in \mathcal{T}} |C| + |D|$.

Let $\mathsf{N_I}$ be a countably infinite set of *individual names*. An ABox $\mathcal{A}$ is a finite non-empty set containing *concept name assertions* $A(a)$ and *role assertions* $r(a, b)$, 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. Assertions of the form $C(a)$ and $r(a, b)$, where $a, b \in \mathsf{N_I}$, C an $\mathcal{ELI}$ concept expression, and $r \in \mathsf{N_R}$, are called *instance assertions*. Note that instance assertions of the form $C(a)$ with C not a concept name nor $C = \top$ do not occur in ABoxes. The semantics of description logic is defined as usual [3]. We write $\mathcal{I} \models \alpha$ to say that an inclusion or assertion α is true in $\mathcal{I}$. An interpretation $\mathcal{I}$ is a *model* of a KB $(\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T} \cup \mathcal{A}$. $(\mathcal{T}, \mathcal{A}) \models \alpha$ means that $\mathcal{I} \models \alpha$ 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*[1] and μ is a mapping from $\mathcal{L}$ to 2^X. The *subsumption* learning framework $\mathfrak{F}_{\mathcal{S}}$, studied in [12], is defined as $(X_{\mathcal{S}}, \mathcal{L}, \mu_{\mathcal{S}})$, where $\mathcal{L}$ is the set of all TBoxes that are formulated in a given DL; $X_{\mathcal{S}}$ is the set of *subsumption examples* of the form $C \sqsubseteq D$, where C, D are concept expressions of the DL under consideration; and $\mu_{\mathcal{S}}(\mathcal{T})$ is defined as $\{C \sqsubseteq D \in X_{\mathcal{S}} \mid \mathcal{T} \models C \sqsubseteq D\}$, for every $\mathcal{T} \in \mathcal{L}$. It should be clear that $\mu_{\mathcal{S}}(\mathcal{T}) = \mu_{\mathcal{S}}(\mathcal{T}')$ if, and only if, the TBoxes $\mathcal{T}$ and $\mathcal{T}'$ entail the same set of inclusions, that is, they are logically equivalent.

[1] In the learning literature (e.g., [1]), the term 'learning concept' is often defined as a set of examples. We do not distinguish between learning concepts and their representations and only consider representable learning concepts to emphasize on the task of identifying a TBox that is logically equivalent to the target TBox.

We study the *data retrieval* learning framework $\mathfrak{F}_{\mathcal{D}}$ defined as $(X_{\mathcal{D}}, \mathcal{L}, \mu_{\mathcal{D}})$, where $\mathcal{L}$ is same as in $\mathfrak{F}_{\mathcal{S}}$; X is the set of *data retrieval examples* of the form $(\mathcal{A}, D(a))$, where $\mathcal{A}$ is an ABox, $D(a)$ is a concept assertion of the DL under consideration, and $a \in \mathsf{Ind}(\mathcal{A})$; and $\mu(\mathcal{T}) = \{(\mathcal{A}, D(a)) \in X_{\mathcal{D}} \mid (\mathcal{T}, \mathcal{A}) \models D(a)\}$. As in the case of learning from subsumptions, $\mu_{\mathcal{S}}(\mathcal{T}) = \mu_{\mathcal{S}}(\mathcal{T}')$ if, and only if, the TBoxes $\mathcal{T}$ and $\mathcal{T}'$ are logically equivalent.

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. A learning framework $(X, \mathcal{L}, \mu)$ is *polynomially* exact learnable if it is exact learnable by an algorithm A such that at every step[2] 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[3]. As argued in the introduction, for learning subsumption and data retrieval learning frameworks we additionally assume that the signature of the target TBox is always known to the learner.

An important class of learning algorithms—in particular, all algorithms presented in [12, 10, 16] fit in this class—always make equivalence queries with hypotheses h which are polynomial in the size of l and such that $\mu(h) \subseteq \mu(l)$, so that counterexamples returned by the $\mathsf{EQ}_{l,X}$ oracles are always positive. We say that such algorithms use *positive bounded equivalence queries*.

3 Polynomial Time Learnability

In this section we prove polynomial time exact learnability of the DL-Lite$^{\exists}_{\mathcal{R}}$ and $\mathcal{EL}_{\mathsf{lhs}}$ data retrieval learning frameworks. These frameworks are instances of the general definition given above, where the concept expression D in a data retrieval example $(\mathcal{A}, D(a))$ is an $\mathcal{ELI}$ concept expression in the DL-Lite$^{\exists}_{\mathcal{R}}$ framework and an $\mathcal{EL}$ concept expression in the $\mathcal{EL}_{\mathsf{lhs}}$ framework, respectively.

The proof is by reduction to learning from subsumptions. We illustrate its idea for $\mathcal{EL}_{\mathsf{lhs}}$. 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

[2] We count each call to an oracle as one step of computation.

[3] We assume some natural notion of a length of an example x and a learning concept l, denoted $|x|$ and $|l|$, respectively.

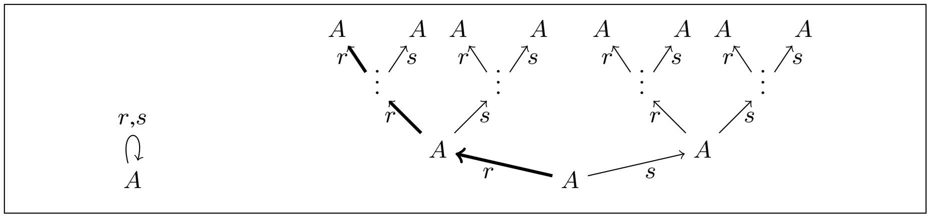

Fig. 1: An ABox $\mathcal{A} = \{r(a, a), s(a, a), A(a)\}$ and its unravelling up to level n.

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 $C \sqsubseteq D$, 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 $\mathcal{T} \models C \sqsubseteq D$ iff $(\mathcal{T}, \mathcal{A}_C) \models D(\rho_C)$.

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 instance queries described above. An inclusion $C \sqsubseteq D$ is a (positive) subsumption example for some target TBox $\mathcal{T}$ if, and only if, $(\mathcal{A}_C, D(\rho_C))$ 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 $(\mathcal{A}, D(a))$ one can construct a concept expression $C_{\mathcal{A}}$ such that $(\mathcal{T}, \mathcal{A}) \models D(a)$ iff $\mathcal{T} \models C_{\mathcal{A}} \sqsubseteq D$. 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 bound on the running time of 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 notation, we assume that both learning frameworks use the same set of learning concepts $\mathcal{L}$ and only consider positive bounded equivalence queries. This definition of reduction can be easily extended to arbitrary learning frameworks and arbitrary queries.

We say that a learning framework $\mathfrak{F} = (X, \mathcal{L}, \mu)$ *polynomially reduces* to $\mathfrak{F}' = (X', \mathcal{L}, \mu')$ if for some polynomials $p_1(\cdot)$, $p_2(\cdot)$ and $p_3(\cdot,\cdot)$ there exist a function $f : X' \to X$ and a partial function $g : \mathcal{L} \times \mathcal{L} \times X \to X'$, defined for every (l, h, x) such that $|h| = p_1(|l|)$, $\mu(h) \subseteq \mu(l)$ and $x \in X$, for which the following conditions hold.

- For all $x' \in X'$ we have $x' \in \mu'(l)$ if, and only if, $f(x') \in \mu(l)$;
- For all $x \in X$ we have $x \in \mu(l) \setminus \mu(h)$ if, and only if, $g(l, h, x) \in \mu'(l) \setminus \mu'(h)$;
- $f(x')$ is computable in time $p_2(|x'|)$;
- $g(l, h, x)$ is computable in time $p_3(|l|, |x|)$ and l can only be accessed by calls to the membership oracle $\mathsf{MEM}_{l,X}$.

As in the case of learning algorithms, we consider every call to the oracle as one step of computation. Notice also that even though g takes h as input, the polynomial time bound on computing $g(l, h, x)$ does not depend on the size of h as g is only defined for h polynomial in the size of l.

Theorem 1. *Let $(X, \mathcal{L}, \mu)$ and $(X', \mathcal{L}, \mu')$ be learning frameworks. If there exists a polynomial reduction from $(X, \mathcal{L}, \mu)$ to $(X', \mathcal{L}, \mu')$ and a polynomial learning algorithm for $(X', \mathcal{L}, \mu')$ that uses membership queries and positive bounded equivalence queries then $(X, \mathcal{L}, \mu)$ is polynomially exact learnable.*

We use Theorem 1 to prove that DL-Lite$^{\exists}_{\mathcal{R}}$ and $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes can be learned in polynomial time from data retrieval examples. We employ the following result:

Theorem 2 ([12]). *The DL-Lite$^{\exists}_{\mathcal{R}}$ and $\mathcal{EL}_{\mathsf{lhs}}$ subsumption learning frameworks are polynomial time exact learnable with membership and positive bounded equivalence queries.*

As the existence of f is guaranteed by the following lemma, in what follows we prove the existence of g and establish the corresponding time bounds.

Lemma 1. *Let $L \in \{$DL-Lite$^{\exists}_{\mathcal{R}}, \mathcal{EL}_{\mathsf{lhs}}\}$ and let $C \sqsubseteq D$ be an L concept inclusion. Then $(\mathcal{T}, \mathcal{A}_C) \models D(\rho_C)$ if, and only if, $\mathcal{T} \models C \sqsubseteq D$.*

Polynomial Reduction for DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes We show for any target $\mathcal{T}$ and hypothesis $\mathcal{H}$ polynomial in the size of $\mathcal{T}$ that Algorithm 1 transforms every positive counterexample in polynomial time to a positive counterexample with a singleton ABox (i.e., of the form $\{A(a)\}$ or $\{r(a, b)\}$). Using the equivalences $(\mathcal{T}, \{A(a)\}) \models C(a)$ iff $\mathcal{T} \models A \sqsubseteq C$ and $(\mathcal{T}, \{r(a, b)\}) \models C(a)$ iff $\mathcal{T} \models \exists r.\top \sqsubseteq C$, we then obtain a positive subsumption counterexample, so $g(l, h, x)$ is computable in polynomial time.

Given a positive data retrieval counterexample $(\mathcal{A}, C(a))$, Algorithm 1 exhaustively applies the *role saturation* and *parent-child merging* rules introduced in [12]. We say that an instance assertion $C(a)$ is *role saturated* for $(\mathcal{T}, \mathcal{A})$ if $(\mathcal{T}, \mathcal{A}) \not\models C'(a)$ whenever C' is the result of replacing a role r by some role $s \in \mathsf{N_R} \cap \Sigma_{\mathcal{T}}$ with $\mathcal{T} \not\models r \sqsubseteq s$ and $\mathcal{T} \models s \sqsubseteq r$, where $\Sigma_{\mathcal{T}}$ is the signature of the target TBox $\mathcal{T}$ known to the learner. 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 in the standard way. For example, if $C = \exists t.(A \sqcap \exists r.\exists r^-.\exists r.B) \sqcap \exists s.\top$ then Fig. 2a illustrates

Algorithm 1 Reducing the positive counterexample

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

T_C. Now, we say that an instance assertion $C(a)$ 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.

We present a run of Algorithm 1 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).

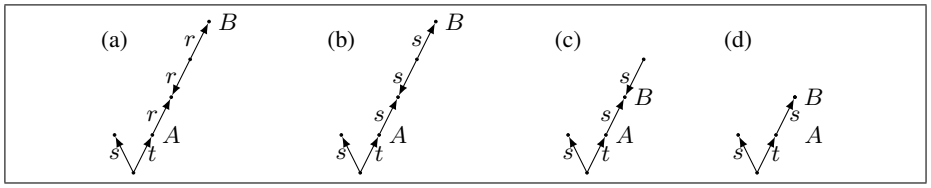

Fig. 2: Concept C being role saturated and parent/child merged.

Since $(\mathcal{H}, \mathcal{A}) \not\models \exists t.(A \sqcap \exists s.B)(a)$, after Lines 4-6, 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 4-6, C is updated to $\exists s.B$. Finally, C is of the form $\exists t.C'$ and there is no $t(b, c) \in \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}) \models C'(c)$. So the algorithm proceeds to Lines 11-12, 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 $\mathcal{H} \not\models A \sqsubseteq \exists s.B$.

Lemma 2. *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 $(\mathcal{T}, \mathcal{A}') \models C(a)$;*

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

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

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

Lemma 3. *For any target DL-Lite$^\exists_\mathcal{R}$ TBox $\mathcal{T}$ and hypothesis DL-Lite$^\exists_\mathcal{R}$ TBox $\mathcal{H}$ given a positive data retrieval counterexample $(\mathcal{A}, C(a))$, Algorithm 1 computes in time polynomial in $|\mathcal{T}|$, $|\mathcal{H}|$, $|\mathcal{A}|$ and $|C|$ a counterexample $C'(b)$ such that $(\mathcal{T}, \mathcal{A}') \models C'(b)$, where $\mathcal{A}' \subseteq \mathcal{A}$ is a singleton ABox.*

Proof. (Sketch) Let $(\mathcal{A}, C(a))$ be the input of "ReduceCounterExample". The number of membership queries in Line 3 is polynomial in $|C|$ and $|\mathcal{T}|$. If C has more than one conjunct then it is updated in Lines 4-6, so C becomes either (1) a basic concept or (2) of the form $\exists r.C'$ (or $\exists r^-.C'$). By Lemma 2 in case (1) there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}') \models C(a)$, computed by Line 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 Line 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 2 and Theorem 1 we obtain:

Theorem 3. *The DL-Lite$^\exists_\mathcal{R}$ data retrieval framework is polynomially exact learnable.*

Polynomial Reduction for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes In this section we give a polynomial algorithm computing g for $\mathcal{EL}_{\mathsf{lhs}}$. First we note that the concept assertion in data retrieval counterexamples $(\mathcal{A}, D(a))$ can always be made atomic. Let $\Sigma_\mathcal{T}$ be the signature of the target TBox $\mathcal{T}$.

Lemma 4. *If $(\mathcal{A}, D(a))$ is a positive counterexample then by posing polynomially many membership queries one can find a concept name $A \in \Sigma_\mathcal{T}$ and an individual $b \in \mathsf{Ind}(\mathcal{A})$ such that $(\mathcal{A}, A(b))$ is also a counterexample.*

Thus it suffices to show that given a positive counterexample $(\mathcal{A}, D(a))$ with $D \in \mathsf{N_C}$, one can compute an $\mathcal{EL}$ concept expression C bounded in size by $|\mathcal{T}|$ such that $(\mathcal{T}, \{C(b)\}) \models A(b)$ and $(\mathcal{H}, \{C(b)\}) \not\models A(b)$, where $A \in \mathsf{N_C}$. As $(\mathcal{T}, \{C(b)\}) \models A(b)$ if and only if $\mathcal{T} \models C \sqsubseteq A$, we obtain a positive subsumption counterexample. Our algorithm for computing g is based on two operations: minimization, computed by

Algorithm 3 Computing a tree shaped ABox

function FINDTREE($\mathcal{A}$)
  MINIMIZEABOX($\mathcal{A}$)
  while there is a cycle c in $\mathcal{A}$ **do**
    Unfold $a \in \mathsf{Ind}(\mathcal{A})$ in cycle c
    MINIMIZEABOX($\mathcal{A}$)
  Let C be the concept expression corresponding to $\mathcal{A}$ with counterexample $A(a)$.
  return $(C(a),A(a))$

Algorithm 2, and unfolding. Algorithm 2 *minimizes* a given ABox with the following rules.

(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 $A(a)$ 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 $A(a)$ 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 $\mathcal{A}^{-r(b,c)}$ be obtained by removing a role assertion $r(b, c)$ from $\mathcal{A}$.

Lemma 5. *Given a positive counterexample $(\mathcal{A}, D(a))$ with $D \in \mathsf{N_C}$, Algorithm 2 computes in polynomially many steps with respect to $|\mathcal{A}|$, $|\mathcal{H}|$, and $|\mathcal{T}|$ an ABox $\mathcal{A}'$ such that $|\mathsf{Ind}(\mathcal{A}')| \leq |\mathcal{T}|$ and $(\mathcal{A}', A(b))$ is a positive counterexample, for some $A \in \mathsf{N_C}$ and $b \in \mathsf{Ind}(\mathcal{A}')$.*

It remains to show that $\mathcal{A}$ can be made tree-shaped. We say that $\mathcal{A}$ has an (undirected) cycle if there is a finite sequence $a_0 \cdot r_1 \cdot a_1 \cdot \ldots \cdot r_k \cdot a_k$ 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 $c = a_0 \cdot r_1 \cdot a_1 \cdot \ldots \cdot r_k \cdot a_k$ in a given ABox $\mathcal{A}$ is obtained by replacing c by the cycle $c' = a_0 \cdot r_1 \cdot a_1 \cdot \ldots \cdot r_k \cdot a_{k-1} \cdot r_k \cdot \widehat{a}_0 \cdot r_1 \cdots \widehat{a}_{k-1} \cdot r_k \cdot a_0$, 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 full version that after every 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 counterexample.

Lemma 6. *Algorithm 3 computes a minimal tree shaped ABox $\mathcal{A}$ with size polynomial in $|\mathcal{T}|$ and runs in polynomially many steps in $|\mathcal{T}|$ and $|\mathcal{A}|$.*

Using Theorem 2 and Theorem 1 we obtain:

Theorem 4. *The $\mathcal{EL}_{\mathsf{lhs}}$ data retrieval framework is polynomially exact learnable.*

4 Limits of Polynomial Time Learnability

Our proof of non-polynomial learnability of general $\mathcal{EL}$ TBoxes from data retrieval examples extends previous results on non-polynomial learnability of $\mathcal{EL}$ TBoxes from

subsumptions [12]. We start by giving a brief overview of the construction in [12], show that it fails in the data retrieval setting and then demonstrate how it can be modified.

The non-learnability proof in [12] 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 $S = \{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$. It has been proved 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, but then the learner cannot distinguish between the remaining TBoxes of our 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:

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

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

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 $X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \dots \sqcap \exists \boldsymbol{\sigma}_n.M \sqsubseteq A \in \mathcal{T}_L$, 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 proceeding with unfolding 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 $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$ is now given by the set $\mathfrak{B}_n$ of all conjunctions $F_1 \sqcap \dots \sqcap F_n$, 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 a TBox $\mathcal{T}^*$ defined as the following set of concept inclusions:

$$\exists r.(E_1 \sqcap \bar{E}_1) \sqsubseteq (E_1 \sqcap \bar{E}_1),\ (E_1 \sqcap \bar{E}_1) \sqsubseteq \exists r.(E_1 \sqcap \bar{E}_1),$$
$$\exists s.(E_1 \sqcap \bar{E}_1) \sqsubseteq (E_1 \sqcap \bar{E}_1),\ (E_1 \sqcap \bar{E}_1) \sqsubseteq \exists s.(E_1 \sqcap \bar{E}_1),$$
$$(E_i \sqcap \bar{E}_i) \sqsubseteq A \quad \text{for every } 1 \leq i \leq n \text{ and every } A \in \Sigma_n \cap \mathsf{N_C}$$

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 7. *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 $\mathbf{B} \in \mathfrak{B}_n$.*

The next lemma is immediate from Lemma 15 presented in [12]. 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 8. *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) the size of $\mathcal{A}$ plus the size of 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 $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$.*

Then, by Lemmas 7 and 8, we have that: (i) any polynomial size membership query can distinguish at most polynomially many TBoxes from S; 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.

Theorem 5. *The $\mathcal{EL}$ data retrieval framework is not polynomially exact learnable.*

5 Future Work

We plan to consider an extension of the learning protocol in which arbitrary conjunctive queries are admitted in queries to the domain expert/oracle. We then still have polynomial time learnability for $\mathcal{EL}_{\mathsf{lhs}}$ but conjecture non-polynomial time learnability for DL-Lite$_{\mathcal{R}}^{\exists}$. Another extension is exact learnability for the Horn-extension of DL-Lite$_{\mathcal{R}}^{\exists}$ for which we conjecture that polynomial time learnability still holds.

Bibliography

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

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

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

[4] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. Oiled: a reason-able ontology editor for the semantic web. In *KI 2001: Advances in Artificial Intelligence*, pages 396–408. Springer, 2001.

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

[6] P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2005.

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

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

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

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

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

[12] B. Konev, C. Lutz, A. Ozaki, and F. Wolter. Exact learning of lightweight description logic ontologies. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, 2014.

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

[14] Y. Ma and F. Distel. Learning formal definitions for snomed CT from text. In *Artificial Intelligence in Medicine - 14th Conference on Artificial Intelligence in Medicine, AIME 2013, Murcia, Spain, May 29 - June 1, 2013. Proceedings*, pages 73–77, 2013.

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

[16] C. Reddy and P. Tadepalli. Learning First-Order Acyclic Horn Programs from Entailment. In *in Proceedings of the 15th International Conference on Machine Learning; (and Proceedings of the 8th International Conference on Inductive Logic Programming*, pages 23–37. Morgan Kaufmann, 1998.

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

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