Exact Learning of TBoxes in EL and DL-Lite

Boris Konev[1] and Carsten Lutz[2] and Frank Wolter[1]

[1] Department of Computer Science, University of Liverpool, UK
[2] Department of Computer Science, University of Bremen, Germany

Abstract. We study polynomial time learning of description logic TBoxes in Angluin et al.'s framework of exact learning via queries. We admit entailment queries ("is a given subsumption entailed by the target TBox?") and equivalence queries ("is a given TBox equivalent to the target TBox?"), assuming that the signature and logic of the target TBox are known. We present three main results: (1) TBoxes formulated in the extension of DL-Lite$_{\mathsf{core}}$ with $\mathcal{ELI}$-concepts on the right-hand side of concept inclusions can be learned in polynomial time. (2) Neither general nor acyclic $\mathcal{EL}$ TBoxes can be learned in polynomial time. (3) $\mathcal{EL}$ TBoxes with only concept names on the right-hand side of concept inclusions can be learned in polynomial time. It follows, in particular, that non-polynomial time learnability of $\mathcal{EL}$ TBoxes is caused by the interaction between existential restrictions on the right and left-hand side of concept inclusions.

1 Introduction

Successfully deploying a description logic (DL) in a concrete application requires to carefully capture the relevant domain knowledge in a DL ontology. This is a subtle, error-prone, and time consuming task, which is further hindered by the fact that domain experts are rarely experts in ontology engineering and, conversely, ontology engineers are often not sufficiently familiar with the domain to be modeled. From its beginnings, DL research was driven by the aim to provide various forms of support for ontology engineers, assisting them in the design of high-quality ontologies. Examples include the ubiquitous task of ontology classification [9], bootstrapping ontology design from examples [11, 12] and checking the completeness of the modeling in a systematic way [10].

Machine learning (ML) techniques are natural candidates for supporting ontology engineering by partly automatizing the design process, and in fact ML approaches have been proposed for ontology engineering in non-DL-contexts. There, the main focus is on mining the relevant terms of the application domain from bodies of text and semi-structured data with the aim of including them in the ontology as concept and role names. Some approaches also support learning of subsumptions between these terms, that is, subsumptions between concept *names* [15, 13]. In the context of DL ontologies, though, the main difficulty is to develop a detailed modeling of the relevent concepts of the domain using the logical operators provided by the DL at hand. The aim of this paper is to investigate the latter problem and to study the theoretical foundations of learning ontologies that are formulated in variations of the description logics $\mathcal{EL}$ and DL-Lite, using Angluin et al.'s framework of exact learning via queries [3].

We are interested in learning a target TBox $\mathcal{T}$ that is formulated in a given DL and whose signature Σ (the concept and role names that occur in $\mathcal{T}$) is also known, by posing queries to an oracle. Intuitively, the oracle can be thought of as a domain expert that interacts with the learning algorithm. We consider two forms of oracle queries:

- *entailment queries*: does $\mathcal{T}$ entail a given Σ-concept inclusion $C \sqsubseteq D$? These queries are answered by the oracle with 'yes' or 'no'.
- *equivalence queries*: is a given Σ-TBox (the hypothesis) equivalent to $\mathcal{T}$? The oracle answers 'yes' if $\mathcal{H}$ and $\mathcal{T}$ are logically equivalent and 'no' otherwise. It then returns a Σ-inclusion $C \sqsubseteq D$ such that $\mathcal{T} \models C \sqsubseteq D$ and $\mathcal{H} \not\models C \sqsubseteq D$ (a *positive counterexample*), or vice versa (a *negative counterexample*).

We generally assume that the inclusions used in entailment queries and returned by equivalence queries are of the same form as the inclusions allowed in the target TBox.

Our aim is to find (deterministic) learning algorithms that run in polynomial time and consequently also make only polynomially many oracle queries. More precisely, we require that there is a two-variable polynomial $p(n, m)$ such that at every point of the algorithm execution, the time spent up to that point is bounded by $p(n, m)$, where n is the size of the target TBox $\mathcal{T}$ to be learned and m is the size of the largest (positive or negative) counterexample that was returned by the oracle until the considered point of the execution. If such an algorithm exists, we say that $\mathcal{L}$-TBoxes are *polynomial time learnable*.

When the target TBox is formulated in a description logic $\mathcal{L}$ such that for any signature Σ, the number of $\mathcal{L}$-concept inclusions is polynomial in the size of Σ, then $\mathcal{L}$-TBoxes are trivially polynomial time learnable. This is actually the case even when only entailment queries (but no equivalence queries) are available, and also when only equivalence queries (but no entailment queries) are available. Consequently, DL-Lite$_{\mathsf{core}}$-TBoxes and DL-Lite$_{\mathcal{R}}$-TBoxes are polynomial time learnable. We thus start our investigation with considering the more interesting case of DL-Lite$_{\exists}$, which is the extension of DL-Lite$_{\mathsf{core}}$ that allows any $\mathcal{ELI}$-concept on the right-hand side of concept inclusions. Although the number of Σ-concept inclusions is no longer polynomial in the size of Σ, we are still able to establish that DL-Lite$_{\exists}$-TBoxes are polynomial time learnable.

We then switch to the case of $\mathcal{EL}$-TBoxes. It is known that propositional Horn formulas (which correspond to $\mathcal{EL}$-TBoxes without existential restrictions) can be learned in polynomial time when both entailment and equivalence queries are available, but not when one of the types of queries is disallowed [16, 4, 1], see also [5]. We show that, in contrast, general $\mathcal{EL}$ TBoxes and even acyclic $\mathcal{EL}$ TBoxes are not polynomial time learnable. To analyze the reasons for this, we consider the fragment $\mathcal{EL}_{\mathsf{lhs}}$ of $\mathcal{EL}$ in which TBoxes consist of concept inclusions with only concept names on the right hand side. It turns out that $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes are polynomial time learnable. Note that the corresponding fragment $\mathcal{EL}_{\mathsf{rhs}}$ that allows only concept names on the left hand side of concept inclusions is a fragment of DL-Lite$_{\exists}$, and thus also admits polynomial time learning. Consequently, non-polynomial time learnability of (acyclic) $\mathcal{EL}$ TBoxes is caused by the interaction between existential restrictions on the right- and left-hand side of concept inclusions.

Missing proofs are given in the appendix of the full version of this paper available at http://cgi.csc.liv.ac.uk/~frank/publ/publ.html

Related Work. In the learning literature, entailment queries are regarded as a special type of membership queries which can take many different forms [21]. Learning using membership and equivalence queries appears to be the most successful protocol for exact learning in Angluin's framework. Apart from propositional Horn formulas, important polynomial time learnable classes for this protocol include regular sets [2] and monotone DNF [3]. Exploring the possibilities of extending the learning algorithm for propositional Horn formulas to (fragments) of first-order Horn logic has been a major topic in exact learning [22, 6]. Note that $\mathcal{EL}_{\mathsf{lhs}}$ can be regarded as a fragment of first-order Horn logic. Existing approaches either disallow recursion, bound the number of individual variables per Horn clause, or admit additional queries in the learning protocol. None of this is the case in our setup. Another topic related to our work is exact learning of schema mappings in data exchange, as recently studied in [23]. In this and some other studies mentioned above, membership queries take the form of interpretations ("is a given interpretation a model of the target theory?"). In the DL context, exact learning has been studied for CLASSIC in [17] where it is shown that single CLASSIC concepts (but not TBoxes) can be learned in polynomial time (here membership queries consist of concepts). Learning single concepts using refinement operators has been studied in [19].

2 Preliminaries

Let $\mathsf{N_C}$ be a countably infinite set of *concept names* and $\mathsf{N_R}$ a countably infinite set of *role names*. We consider a dialect of DL-Lite that we call DL-Lite$_\exists$, defined as follows [14]. A *role* is a role name or an inverse role r^- with $r \in \mathsf{N_R}$. A *basic concept* is either a concept name or of the form $\exists r.\top$, with r a role. A *DL-Lite$_\exists$ concept inclusion (CI)* is of the form $B \sqsubseteq C$, where B is a basic concept and C is an $\mathcal{ELI}$-concept, that is, C is formed according to the rule

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

where A ranges over $\mathsf{N_C}$ and r ranges over $\mathsf{N_R}$. A *DL-Lite$_\exists$ TBox* is a finite set of DL-Lite$_\exists$ inclusions. As usual, an *$\mathcal{EL}$-concept* is an $\mathcal{ELI}$-concept that does not use inverse roles, an *$\mathcal{EL}$ concept inclusion* has the form $C \sqsubseteq D$ with C and D $\mathcal{EL}$-concepts, and a *(general) $\mathcal{EL}$ TBox* is a finite set of $\mathcal{EL}$ concept inclusions [8]. We use $C \equiv D$ as an abbreviation for the inclusions $C \sqsubseteq D$ and $D \sqsubseteq C$. An $\mathcal{EL}$ TBox $\mathcal{T}$ is called *acyclic* if it consists of inclusions $A \sqsubseteq C$ and $A \equiv C$ such that A is a concept name, no concept names occurs more than once on the left hand side of an inclusion in $\mathcal{T}$, and $\mathcal{T}$ contains no cycles [18].

A *signature* Σ is a finite set of concept and role names. The *size* of a concept 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. The *size* of a TBox $\mathcal{T}$, denoted with $|\mathcal{T}|$, is $\sum_{C \sqsubseteq D \in \mathcal{T}} |C| + |D|$.

3 DL-Lite$_\exists$-TBoxes are Learnable in Polynomial Time

We prove that DL-Lite$_\exists$-TBoxes can be learned in polynomial time.[3] We assume that the target TBox is in *named form*, that is, it contains for each role r a concept name A_r such that $\exists r.\top \sqsubseteq A_r \in \mathcal{T}$ and $A_r \sqsubseteq \exists r.\top \in \mathcal{T}$. This assumption is without loss of generality since any (polynomial time) learning algorithm for TBoxes in named form can be transformed into one for unrestricted TBoxes: the learning algorithm still uses the concept names A_r in its internal representations (although they are no longer included in the target signature Σ), and replaces each A_r with $\exists r.\top$ in queries to the oracle and when ultimately returning the TBox that is has learned.

As a first step, the learning algorithm for DL-Lite$_\exists$ TBoxes determines the set of all inclusions $B_1 \sqsubseteq B_2$ with $\mathcal{T} \models B_1 \sqsubseteq B_2$ and B_1, B_2 basic concepts. Since the target signature Σ is known to the learner, this requires at most $(2|\Sigma|)^2$ entailment queries. We write $B_1 \equiv_\mathcal{T} B_2$ when $\mathcal{T} \models B_1 \sqsubseteq B_2$ and $\mathcal{T} \models B_2 \sqsubseteq B_1$. After completing the first step, the learning algorithm fixes for each $[B]_\mathcal{T} = \{B' \mid B \equiv_\mathcal{T} B'\}$ a representative $A_{[B]_\mathcal{T}} \in [B]_\mathcal{T}$ which we may assume to be a concept name since $\mathcal{T}$ is in named form. A DL-Lite$_\exists$ inclusion $A \sqsubseteq C$ is in *reduced form* if all basic concepts that occur in it are among the chosen representatives. We assume without further notice that all concept inclusions considered by the learner are in reduced form. In particular, counterexamples returned by the oracle are immediately converted into this form.

For a transparent formulation of the learning algorithm, it is useful to identify each $\mathcal{ELI}$ concept C with a finite tree T_C whose nodes are labeled with sets of concept names and whose edges are labeled with roles. Specifically, T_C has a root node a_C and non-root nodes a_D for every occurrence of a subconcept $\exists r.D$ of C; each node a_D is labeled with the set of concept names $l(a_D) = \{A \mid A \text{ is a top-level conjunct of } D\}$ and every edge $(a_{D \sqcap \exists r.D'}, a_{D'})$ is labeled with the role $\{r\}$. Conversely, every tree T of the described form gives rise to an $\mathcal{ELI}$ concept C_T in the obvious way. We will not always distinguish explicitly between C and its tree representation T_C and talk, for example, about the nodes and subtrees of an $\mathcal{ELI}$ concept.

The following notion plays a central role in the learning algorithm. Let $\mathcal{T}$ be a DL-Lite$_\exists$ TBox. A DL-Lite$_\exists$ inclusion $A \sqsubseteq C$ is *$\mathcal{T}$-essential* if it is in reduced form, $\mathcal{T} \models A \sqsubseteq C$, and the following conditions are satisfied:

1. $A \sqsubseteq C$ is *concept saturated for $\mathcal{T}$*: if C' results from C by adding a concept name A' to the label of some node, then $\mathcal{T} \not\models A \sqsubseteq C'$.
2. $A \sqsubseteq C$ is *sibling-merged for $\mathcal{T}$*: if C' is the result of identifying two siblings in C, then $\mathcal{T} \not\models A \sqsubseteq C'$.
3. $A \sqsubseteq C$ is *parent/child-merged for $\mathcal{T}$*: if C' is the result of identifying a parent and a child of a node in the tree representation of C, then $\mathcal{T} \not\models A \sqsubseteq C'$.
4. $A \sqsubseteq C$ is *minimal for $\mathcal{T}$*: if C contains an edge (d, d') such that A' is in the node label of d and $l(d, d') = r$, then $\mathcal{T} \not\models A' \sqsubseteq \exists r.C'$ where C' corresponds to the subtree rooted at d' (where $A' \neq A$ if d is the root of C).

[3] We assume that the CIs used in entailment and in equivalence queries and those returned as counterexamples by the oracle are also formulated in DL-Lite$_\exists$. It is also conceivable to use more expressive logics instead.

Algorithm 1 The learning algorithm for DL-Lite$_\exists$

1: Compute $\mathcal{H}_{basic} = \{B_1 \sqsubseteq B_2 \mid \mathcal{T} \models B_1 \sqsubseteq B_2,\ B_1, B_2 \text{ basic}\}$ (entailment queries)
2: Let $\mathcal{H}_{add} = \emptyset$
3: **while** Equivalent($\mathcal{H}_{basic} \cup \mathcal{H}_{add}$)? returns "no" **do**
4: Let $B \sqsubseteq D$ be the returned positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$
5: Find a $\mathcal{T}$-essential inclusion $A \sqsubseteq C$ with $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A \sqsubseteq C$ (entailment queries)
6: **if** there is $A \sqsubseteq C' \in \mathcal{H}_{add}$ **then**
7: Find $\mathcal{T}$-essential inclusion $A \sqsubseteq C''$ such that $\models C'' \sqsubseteq C \sqcap C'$ (entailment queries)
8: Replace $A \sqsubseteq C'$ by $A \sqsubseteq C''$ in $\mathcal{H}_{add}$
9: **else**
10: add $A \sqsubseteq C$ to $\mathcal{H}_{add}$
11: **end if**
12: **end while**
13: Set $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$.

In Points 2 and 3, we only consider the merging of nodes that are reachable via the same role. In Point 3, for example, C' is the result of selecting two edges (d_1, d) and (d, d_2) such that $l(d_1, d)$ is the inverse of $l(d, d_2)$ and merging d_1 with d_2, labeling the resulting node with $l(d_1) \cup l(d_2)$. The algorithm for learning DL-Lite$_\exists$ TBoxes is given as Algorithm 1. In the remainder of this section, we prove that this algorithm is indeed capable of learning DL-Lite$_\exists$ TBoxes in named form, and that it runs in polynomial time. First observe that in Line 4 the assumption that a positive counterexample is returned by the oracle (i.e., an inclusion entailed by $\mathcal{T}$ but not by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$) is justified by the construction of $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ in Lines 1, 8, and 10 which ensure that $\mathcal{T} \models \mathcal{H}_{basic} \cup \mathcal{H}_{add}$. We now show that Lines 5 and 7 can be implemented using only polynomially many entailment queries. In the next lemma, we consider Line 5.

Lemma 1. *Given a positive counterexample $A \sqsubseteq C$ for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$, one can construct a $\mathcal{T}$-essential $A' \sqsubseteq C'$ with $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A' \sqsubseteq C'$ using polynomially many entailment queries in $|C|$ and $|\mathcal{T}|$.*

Proof. Assume the DL-Lite$_\exists$ concept inclusion $A \sqsubseteq C$ is given. We may assume that $A \sqsubseteq C$ is in reduced form and exhaustively apply the following rules, which rely on posing entailment queries to the oracle:

(Saturation) if $\mathcal{T} \models A \sqsubseteq C'$ and C' results from C by adding a concept name A' to the label of some node, then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Sibling merging) if $\mathcal{T} \models A \sqsubseteq C'$ and C' is the result of identifying two siblings in C, then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Parent/child merging) if $\mathcal{T} \models A \sqsubseteq C'$ and C' is the result of identifying a parent and a child of a node in C, then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Minimization) if $\mathcal{T} \models A' \sqsubseteq \exists r.C'$ and $A' \sqsubseteq \exists r.C'$ is obtained from $A \sqsubseteq C$ by selecting an edge (d, d') in C such that A' is in the node label of d, $l(d, d') = r$, and C' corresponds to the subtree rooted at d', and $A' \neq A$ if d is the root of C, then replace $A \sqsubseteq C$ by

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

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

In the first three rules, we have $\mathcal{T} \models A \sqsubseteq C'$ and $\models C' \sqsubseteq C$ if $A \sqsubseteq C$ is replaced by $A \sqsubseteq C'$. Hence $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A \sqsubseteq C'$. In the last rule, observe that

$$\{A' \sqsubseteq \exists r.C', A \sqsubseteq C|_{d'\downarrow}\} \models A \sqsubseteq C.$$

Hence, independently of whether (a) or (b) applies, the result of the rule application is entailed by $\mathcal{T}$ and not entailed by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. It follows directly from the definition of the rules that the final concept inclusion is $\mathcal{T}$-essential. Clearly, the number of rule applications is bounded by $|C| \times |\Sigma|$. (Note that if d is the root of C in the last rule, then $\mathcal{T} \models A \sqsubseteq A'$ and $\mathcal{T} \not\models A' \sqsubseteq A$ because concept inclusions are in reduced form; thus, this rule does not lead to non-termination of rule application.) ❑

The following lemma addresses Line 7 and is proved in the appendix.

Lemma 2. *Assume that $A \sqsubseteq C_1$ and $A \sqsubseteq C_2$ are $\mathcal{T}$-essential. Then one can construct a $\mathcal{T}$-essential $A \sqsubseteq C$ such that $\models C \sqsubseteq C_1 \sqcap C_2$ using polynomially many entailment queries in $|C_1| + |C_2|$.*

If the algorithm terminates, then it obviously terminates with a hypothesis $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ that is logically equivalent to $\mathcal{T}$. It thus remains to be shown that the algorithm runs in polynomial time. Observe that $\mathcal{H}_{add}$ contains at most one concept inclusion $A \sqsubseteq C$ for each concept name A. At each step in the while-loop either a fresh $A \sqsubseteq C$ is added to $\mathcal{H}_{add}$ (if no inclusion with A on the left exists in $\mathcal{H}_{add}$) or one such $A \sqsubseteq C$ is replaced by a fresh $A \sqsubseteq C'$ with $\models C' \sqsubseteq C$ and $\not\models C \sqsubseteq C'$. Termination in polynomial time thus follows if the number of replacements of one $A \sqsubseteq C$ by a fresh $A \sqsubseteq C'$ is bounded by a polynomial. It follows from the condition that $A \sqsubseteq C$ and $A \sqsubseteq C'$ are $\mathcal{T}$-essential that the tree corresponding to C is obtained from the tree correponding to C' by removing subtrees. Now termination in polynomial time follows from the following lemma which is proved in the appendix using canonical models for DL-Lite$_\exists$-TBoxes. Let $n(C)$ denote the number of nodes in the tree representation of C.

Lemma 3. *If $A \sqsubseteq C$ is $\mathcal{T}$-essential, then $n(C) \leq \sum_{A \sqsubseteq D \in \mathcal{T}} n(D)$.*

It is proved in the appendix that all four properties of $\mathcal{T}$-essential inclusions are necessary to ensure that Algorithm 1 is correct and polynomial time. We formulate the main result of this section.

Theorem 1. *DL-Lite$_\exists$ TBoxes are polynomial time learnable using entailment and equivalence queries.*

4 $\mathcal{EL}$ TBoxes are not Polynomial Time Learnable

We show that acyclic $\mathcal{EL}$ TBoxes cannot be learned in polynomial time. Consequently, the same is true for general $\mathcal{EL}$ TBoxes. Although the target TBox is acyclic, we assume that CIs used in entailment queries and in equivalence queries and those returned as

counterexamples are formulated in unrestricted $\mathcal{EL}$; in particular, compound concepts are allowed both on the left- and right-hand side.

The proof is inspired by Angluin's lower bound for the following abstract learning problem [3]: a learner aims to identify one of N distinct sets $L_1, \dots, L_N$ which have the property that there exists a set $L_\cap$ for which $L_i \cap L_j = L_\cap$, for any $i \neq j$. It is assumed that $L_\cap$ is not a valid argument to an equivalence query. The learner can pose membership queries "$x \in L$?" and equivalence queries "$H = L$?". Then in the worst case it takes at least $N - 1$ membership and equivalence queries to exactly identify a hypothesis L_i from $L_1, \dots, L_N$. The proof proceeds as follows. At every stage of computation the oracle maintains a set of hypotheses S, which the learner is not able to distinguish based on the answers given so far. Initially, $S = \{L_1, \dots, L_N\}$. When the learner asks a membership query x, the oracle returns 'Yes' if $x \in L_\cap$ and 'No' otherwise. In the latter case, the (unique) L_i such that $x \in L_i$ is removed from S. When the learner asks an equivalence query H, the oracle returns 'No' and a counterexample $x \in L_\cap \oplus H$ (the symmetric difference of $L_\cap$ and H). This always exists as $L_\cap$ is not a valid query. If the counterexample x is not a member of $L_\cap$, (at most one) $L_i \in S$ such that $x \in L_i$ is eliminated from S. In the worst case, the learner has to reduce the cardinality of S to one to exactly identify a hypothesis, which takes $N - 1$ queries.

Similarly to the method outlined above, in our proof of Theorem 2 we maintain a set of acyclic $\mathcal{EL}$ TBoxes S whose members the learning algorithm is not able to distinguish based on the answers obtained so far. We start with $S = \{\mathcal{T}_1, \dots, \mathcal{T}_N\}$, where N is superpolynomial in the sizes of every TBox $\mathcal{T}_i$, and describe an oracle that responds to entailment and equivalence queries. For didactic purposes, we first present a set of acyclic TBoxes $\mathcal{T}_1, \dots, \mathcal{T}_N$, for which the oracle can respond to entailment queries in the way described above but which is polynomial time learnable when equivalence queries are also allowed. We then show how the TBoxes can be modified to obtain a family of acyclic TBoxes that is not polynomial time learnable using entailment and equivalence queries.

To present the TBoxes, we introduce the following abbreviation. If $n > 0$ and $\boldsymbol{\sigma} = \sigma^1\sigma^2 \dots \sigma^n$ is a sequence of role names where every σ^j is either the role name r or the role name s, and C is a concept, then the expression $\exists\boldsymbol{\sigma}.C$ stands for $\exists\sigma^1.\exists\sigma^2 \dots \exists\sigma^n.C$. There are $N = 2^n$ different sequences $\boldsymbol{\sigma}$ of length n. For every such sequence $\boldsymbol{\sigma}$, consider the acyclic $\mathcal{EL}$ TBox $\mathcal{T}_\sigma$ defined as

$$\mathcal{T}_{\boldsymbol{\sigma}} = \{A \sqsubseteq \exists\boldsymbol{\sigma}.M \sqcap X_0\} \cup \mathcal{T}_0 \text{ with}$$
$$\mathcal{T}_0 = \{X_0 \sqsubseteq \exists r.X_1 \sqcap \exists s.X_1, X_1 \sqsubseteq \exists r.X_2 \sqcap \exists s.X_2, \dots, X_{n-1} \sqsubseteq \exists r.X_n \sqcap \exists s.X_n\}$$

where $\mathcal{T}_0$ generates a full binary tree with edges labelled with role names r and s and with X_0 at the root, X_1 at level 1 and so on; and M is a concept name that 'marks' a particular path in this tree given by the sequence $\boldsymbol{\sigma}$. One can use Angluin's strategy to show that TBoxes from the set S of all such TBoxes $\mathcal{T}_{\boldsymbol{\sigma}}$ cannot be learned in polynomial time using entailment queries: notice that for no sequence $\boldsymbol{\sigma}' \neq \boldsymbol{\sigma}$ of length n we have $\mathcal{T}_{\boldsymbol{\sigma}} \models A \sqsubseteq \exists\boldsymbol{\sigma}'.M$. Thus, following Angluin's strategy, an entailment query of the form $A \sqsubseteq \exists\boldsymbol{\sigma}.M$ eliminates at most one TBox from the set of TBoxes that the learner cannot distinguish. This observation can be generalized to arbitary entailment queries since one can prove, similarly to the proof of Lemma 4, for any $C \sqsubseteq D$ that either

$\{A \sqsubseteq X_0\} \cup \mathcal{T}_0 \models C \sqsubseteq D$ or at most one TBox from S entails $C \sqsubseteq D$. It follows that acyclic $\mathcal{EL}$ TBoxes are not polynomial time learnable using entailment queries only.

Unfortunately, a single equivalence query is sufficient to learn any TBox from S in two steps: given the equivalence query $\{A \sqsubseteq X_0\} \cup \mathcal{T}_0$, the oracle has no other option but to reveal the target TBox $\mathcal{T}_{\boldsymbol{\sigma}}$ as $A \sqsubseteq \exists \boldsymbol{\sigma}.M$ can be found 'inside' every counterexample. Our strategy to avoid this kind of equivalence queries is to modify $\mathcal{T}_1, \dots, \mathcal{T}_N$ in such a way that even though a TBox $\mathcal{T}_\cap$ axiomatizing the intersection over the set of consequences of each $\mathcal{T}_i$, $i \leq N$, exists, its size is exponential and so cannot be used as an equivalence query by a polynomial learning algorithm.

For every $n > 0$ and every n-tuple $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, where every $\boldsymbol{\sigma}_i$ is a role sequence of length n, we define an acyclic $\mathcal{EL}$ TBox $\mathcal{T}_L$ as the union of $\mathcal{T}_0$ and the following inclusions:[4]

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

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

Σ_n denotes the signature of L. Let $\mathfrak{L}_n$ be a set of n-tuples such that for every $1 \leq i \leq n$ and every $L, L' \in \mathfrak{L}_n$ with $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, $L' = (\boldsymbol{\sigma}'_1, \dots, \boldsymbol{\sigma}'_n)$, if $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}'_j$ then $L = L'$ and $i = j$. Then for any sequence $\boldsymbol{\sigma}$ of length n there exists a unique $L \in \mathfrak{L}_n$ and a unique $i \leq n$ such that $\mathcal{T}_L \models A_i \sqsubseteq \exists \boldsymbol{\sigma}.M$ and $\mathcal{T}_L \models B_i \sqsubseteq \exists \boldsymbol{\sigma}.M$. There are $N = \lfloor 2^n/n \rfloor$ different tuples in $\mathfrak{L}_n$. Notice that N is superpolynomial in the size of $\mathcal{T}_L$, for every $L \in \mathfrak{L}_n$.

Every $\mathcal{T}_L$, for $L \in \mathfrak{L}_n$, entails, among other inclusions, $\bigsqcap_{i=1}^n C_i \sqsubseteq A$, where every C_i is either A_i or B_i. There are 2^n different such inclusions, which indicates that the size of the 'intersection TBox' requires superpolynomially many axioms. It follows from Lemma 5 below that this is indeed the case.

The following lemma (proved in the appendix) enables us to respond to entailment queries without eliminating too many TBoxes from the list S of TBoxes that the learner cannot distinguish.

Lemma 4. *For every $\mathcal{EL}$ concept inclusion $C \sqsubseteq D$ over Σ_n:*

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

To illustrate the result consider two TBoxes $\mathcal{T}_L$ and $\mathcal{T}_{L'}$, where $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$ and $L' = (\boldsymbol{\sigma}'_1, \dots, \boldsymbol{\sigma}'_n)$. Then the inclusion $X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \exists \boldsymbol{\sigma}'_1.M \sqcap A_2 \sqcap \dots \sqcap A_n \sqsubseteq A$ is entailed by both $\mathcal{T}_L$ and $\mathcal{T}_{L'}$ but not by any other $\mathcal{T}_{L''}$.

Now we turn our attention to equivalence queries. We show that given any polynomial size equivalence query $\mathcal{H}$ the oracle can return a counterexample inclusion $C \sqsubseteq D$ such that either (i) $\mathcal{H} \models C \sqsubseteq D$ and $\mathcal{T}_L \models C \sqsubseteq D$ for at most one $L \in \mathfrak{L}_n$ or (ii) $\mathcal{H} \not\models C \sqsubseteq D$ and for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L \models C \sqsubseteq D$. Thus, such a counterexample eliminates at most one $\mathcal{T}_L$ from the set S of TBoxes the learner cannot distinguish.

[4] In fact, to prove non-polynomial learnability, it suffices to consider $\exists \boldsymbol{\sigma}_1.M \sqcap \dots \sqcap \exists \boldsymbol{\sigma}_n.M \sqsubseteq A$ in place of the concept equality; however, inclusions of this form are not allowed in acyclic TBoxes.

In addition we have to take extra care of the size of counterexamples as the learning algorithm is allowed to run in time polynomial not only in the size of the target TBox but also in the size of the largest counterexample to equivalence queries returned by the oracle. For instance, if the hypothesis TBox $\mathcal{H}$ contains an inclusion $C \sqsubseteq D$, which is not entailed by any $\mathcal{T}_L$, one cannot simply return $C \sqsubseteq D$ as a counterexample as the learner will be able to 'pump up' its running time by posing a sequence of equivalence queries with spurious $\mathcal{H}_i = \{C_i \sqsubseteq D_i\}$ as an argument such that the size of $C_{i+1} \sqsubseteq D_{i+1}$ is double the size of $C_i \sqsubseteq D_i$. Then at every stage in a run of the learning algorithm, the running time will be polynomial in the size of the input and the size of the largest counterexample received so far, but the overall running time will be exponential in the size of input. The following lemma addresses this issue.

Lemma 5. *For any $n > 0$ 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 $\mathcal{T}_L \models C \sqsubseteq D$ 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 $\mathcal{T}_L \models C \sqsubseteq D$.*

The following is the main result of this section.

Theorem 2. *Neither general $\mathcal{EL}$ TBoxes nor acyclic $\mathcal{EL}$ TBoxes are polynomial time learnable using entailment and equivalence queries.*

Proof. Assume that acyclic $\mathcal{EL}$ TBoxes are polynomial time learnable. Then there exists a learning algorithm with running time at any stage bounded by a polynomial $p(n, m)$. Choose n such that $\lfloor 2^n/n \rfloor > (p(n, 6n))^2$ and let $S = \mathfrak{L}_n$. We follow Angluin's strategy of removing TBoxes from S in such a way that the learner cannot distinguish between any of the remaining TBoxes. Given an entailment query $C \sqsubseteq D$, if $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$, the answer is 'yes'; otherwise the answer is 'no' and all $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq D$ are removed from S. Given an equivalence query $\mathcal{H}$ the answer is 'no', a counterexample $C \sqsubseteq D$ guaranteed by Lemma 5 is produced, and (at most one) $\mathcal{T}_L$ such that $\mathcal{T}_L \models C \sqsubseteq D$ is removed from S.

As all counterexamples produced are smaller than $6n$, the overall running time of the algorithm is bounded by $p(n, 6n)$. Hence, the learner asks not more than $p(n, 6n)$ queries and the size of every query does not exceed $p(n, 6n)$. By Lemmas 4 and 5, at most $(p(n, 6n))^2$ TBoxes are removed from S during the run of the algorithm. But then the algorithm cannot distinguish between any remaining TBoxes based on the given answers and we have derived a contradiction. ❑

5 $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes are Polynomial Time Learnable

We 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. As in Section 3, we assume that CIs used in entailment queries and in equivalence queries and those returned as counterexamples are also of this restricted form. Our algorithm extends the polynomial time agorithm for learning propositional Horn theories from [4].

We first introduce some notation. An interpretation $\mathcal{I}$ is a *tree interpretation* if the directed graph $(\Delta^{\mathcal{I}}, \bigcup_{r \in \mathsf{N_R}} r^{\mathcal{I}})$ is a tree and $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$ for all distinct $r, s \in \mathsf{N_R}$.

Algorithm 2 The learning algorithm for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes

1: Let $\mathfrak{I}$ be the empty sequence (of finite tree interpretations)
2: Let $\mathcal{H} = \emptyset$ be the empty hypothesis
3: **while** Equivalent($\mathcal{H}$)? returns "no" **do**
4: Let $C \sqsubseteq A$ be the returned positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}$
5: Find an essential $\mathcal{T}$-countermodel $\mathcal{I}$ with $\mathcal{I} \models \mathcal{H}$ (entailment queries)
6: **if** there is a $\mathcal{J} \in \mathfrak{I}$ such that $\mathcal{J} \not\Rightarrow (\mathcal{I} \times_{\mathsf{r}} \mathcal{J})$ and $\mathcal{I} \times_{\mathsf{r}} \mathcal{J} \not\models \mathcal{T}$ **then**
7: Let $\mathcal{J}$ be the first such element of $\mathfrak{I}$
8: Find an essential $\mathcal{T}$-countermodel $\mathcal{J}' \subseteq \mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ (entailment queries)
9: replace $\mathcal{J}$ in $\mathfrak{I}$ with $\mathcal{J}'$
10: **else**
11: append $\mathcal{I}$ to $\mathfrak{I}$
12: **end if**
13: Construct $\mathcal{H} = \{C_{\mathcal{I}} \sqsubseteq A \mid \mathcal{I} \in \mathfrak{I}, \mathcal{T} \models C_{\mathcal{I}} \sqsubseteq A\}$ (entailment queries)
14: **end while**

We generally denote the root of a tree interpretation $\mathcal{I}$ with $\rho_{\mathcal{I}}$. 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 $e \in A^{\mathcal{J}}$, and $((d, e), (d', e')) \in r^{\mathcal{I}\times\mathcal{J}}$ if $(d, d') \in r^{\mathcal{I}}$ and $(e, e') \in r^{\mathcal{J}}$, for any concept name A and role name r. Products preserve the truth of $\mathcal{EL}$ concept inclusions [20] and the product of tree interpretations is a disjoint union of tree interpretations. If $\mathcal{I}$ and $\mathcal{J}$ are tree interpretations, we denote by $\mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ the tree interpretation contained in $\mathcal{I} \times \mathcal{J}$ rooted in $(\rho_{\mathcal{I}}, \rho_{\mathcal{J}})$. Let $\mathcal{I}, \mathcal{J}$ be interpretations, $d \in \Delta^{\mathcal{I}}$ and $e \in \Delta^{\mathcal{J}}$. A relation $\sim \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ is a *simulation from* $(\mathcal{I}, d)$ *to* $(\mathcal{J}, e)$ if the following conditions are satisfied [20]: (i) $d \sim e$; (ii) $d \in A^{\mathcal{I}}$ and $d \sim e$ implies $e \in A^{\mathcal{J}}$; (iii) $(d, d') \in r^{\mathcal{I}}$ and $d \sim e$ implies $d' \sim e'$ for some $e' \in \Delta^{\mathcal{J}}$ with $(e, e') \in r^{\mathcal{J}}$. We write $(\mathcal{I}, d) \Rightarrow (\mathcal{J}, e)$ to indicate that there is a simulation from $(\mathcal{I}, d)$ to $(\mathcal{J}, e)$. For an $\mathcal{EL}$-concept C, we write $\mathcal{I}_C$ to denote the tree interpretation that is obtained by viewing C as an interpretation in the natural way. Similarly, given a tree interpretation $\mathcal{I}$ we denote by $C_{\mathcal{I}}$ the $\mathcal{EL}$-concept obtained by viewing $\mathcal{I}$ as a concept. An interpretation $\mathcal{I}$ is a $\mathcal{T}$*-countermodel* if $\mathcal{I} \not\models \mathcal{T}$. For a tree interpretation $\mathcal{I}$, let $\mathcal{I}|^{-}_{\mathsf{root}}$ denote the interpretation obtained from $\mathcal{I}$ by removing the root $\rho_{\mathcal{I}}$ of $\mathcal{I}$. For any element of $\mathcal{I}$, let $\mathcal{I}|^{-}_{d\downarrow}$ be $\mathcal{I}$ with the subtree rooted at d removed. A $\mathcal{T}$-countermodel is *essential* if the following conditions are satisfied:

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

Note that there is a close connection between essential $\mathcal{T}$-countermodels and positive counterexamples for $\mathcal{T}$ relative to $\mathcal{H}$. More precisely, if $\mathcal{T}$ is a target TBox and $\mathcal{H}$ a hypothesis, then every essential $\mathcal{T}$-countermodel $\mathcal{I}$ with $\mathcal{I} \models \mathcal{H}$ gives rise to at least one positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}$ of the form $C_{\mathcal{I}} \sqsubseteq A$, where A is a concept name and $\rho_{\mathcal{I}} \notin A^{\mathcal{I}}$. Conversely, we will see that one can extract, from every positive counterexample $C \sqsubseteq A$ for $\mathcal{T}$ relative to $\mathcal{H}$, an essential $\mathcal{T}$-countermodel $\mathcal{I}$ with $\mathcal{I} \models \mathcal{H}$.

The algorithm for learning $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes is given as Algorithm 2. In Line 6, we write $\mathcal{J} \not\Rightarrow (\mathcal{I} \times_{\mathsf{r}} \mathcal{J})$ as shorthand for $(\mathcal{J}, \rho_{\mathcal{J}}) \not\Rightarrow (\mathcal{I} \times_{\mathsf{r}} \mathcal{J}, (\rho_{\mathcal{I}}, \rho_{\mathcal{J}}))$. Note that

the assumption in Line 4 that a *positive* counterexample is returned is justified by the construction of $\mathcal{H}$ in Lines 2 and 13, which ensures that, at all times, $\mathcal{T} \models \mathcal{H}$. In the following, we provide additional details on how to realize the three lines marked with "(entailment queries)". Line 13 is easiest: We simply use entailment queries to find all CIs $C_{\mathcal{I}} \sqsubseteq A$ with $\mathcal{I} \in \mathfrak{I}$ and A a concept name from Σ. We will later show that the length of $\mathfrak{I}$ is bounded polynomially in $|\mathcal{T}|$, therefore polynomially many entailment queries suffice. Lines 5 and 9 are addressed by Lemmas 6 and 7 below.

Lemma 6. *Given a positive counterexample $C \sqsubseteq A$ for $\mathcal{T}$ relative to $\mathcal{H}$, one can construct an essential $\mathcal{T}$-countermodel $\mathcal{I}$ with $\mathcal{I} \models \mathcal{H}$ using only polynomially many entailment queries in $|\mathcal{T}| + |C|$.*

The proof of Lemma 6 starts with selecting as $\mathcal{I}$ a suitable subtree of the interpretation $\mathcal{I}_C$ and then repeatedly removes subtrees from $\mathcal{I}$ until an essential $\mathcal{T}$-countermodel is found. Details are presented in the appendix, and so is the proof of the subsequent lemma.

Lemma 7. *Given essential $\mathcal{T}$-countermodels $\mathcal{I}$ and $\mathcal{J}$ with $\mathcal{I} \times \mathcal{J} \not\models \mathcal{T}$, one can construct an essential $\mathcal{T}$-countermodel $\mathcal{J}' \subseteq \mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ using only polynomially many entailment queries in $|\mathcal{T}| + |\mathcal{I}| + |\mathcal{J}|$. Specifically, $\mathcal{J}'$ is obtained from $\mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ by removing subtrees.*

If the algorithm terminates, then it obviously returns a TBox $\mathcal{H}$ that is equivalent to the TBox $\mathcal{T}$ to be learned. It thus remains to prove that the algorithm terminates after polynomially many steps. Specifically, we show in the appendix to this paper:

(i) the length of the sequence $\mathfrak{I}$ is bounded by the number of CIs in $\mathcal{T}$ and
(ii) each position of the sequence $\mathfrak{I}$ of interpretations is replaced only $|\mathcal{T}| + |\mathcal{T}|^2$ often with a new interpretation.

We note that for the proof of (ii) it is crucial to show that $|\Delta^{\mathcal{I}}| \leq |\mathcal{T}|$ for any essential $\mathcal{T}$-countermodel $\mathcal{I}$.

Theorem 3. *$\mathcal{EL}_{\mathsf{lhs}}$ TBoxes are polynomial time learnable using entailment and equivalence queries.*

6 Future Work

We plan to explore polynomial time learnability of extensions of the languages considered in this paper. The most important such languages are the extension of $\mathcal{EL}_{\mathsf{lhs}}$ with inverse roles (which corresponds to the OWL profile OWL2 RL) and the extension of DL-Lite$_\exists$ by role inclusions and other standard constructors of the DL-Lite family [14, 7]. It is also an interesting open question whether DL-Lite$_\exists$ TBoxes can be learned in polynomial time using equivalence queries only.

Another research direction is the admission of different types of membership queries and counterexamples in the learning protocol. Important examples include using interpretations rather than concept inclusions or, in an OBDA context, certain answers of ABoxes and the target TBox to queries. Our results provide a good starting point for studying such variations. Finally, it would be of interest to explore the consequences of our results for PAC learning of DL TBoxes.

References

1. D. Angluin. Learning propositional Horn sentences with hints. Technical report, Yale University, 1987.
2. D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.
3. D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.
4. D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.
5. M. Arias and J. L. Balcázar. Construction and learnability of canonical Horn formulas. *Machine Learning*, 85(3):273–297, 2011.
6. M. Arias and R. Khardon. Learning closed Horn expressions. *Inf. Comput.*, 178(1):214–240, 2002.
7. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *J. Artif. Intell. Res. (JAIR)*, 36:1–69, 2009.
8. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, pages 364–369. Professional Book Center, 2005.
9. F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.
10. F. Baader, B. Ganter, B. Sertkaya, and U. Sattler. Completing description logic knowledge bases using formal concept analysis. In *IJCAI*, pages 230–235, 2007.
11. F. Baader and R. Molitor. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In *ICCS*, pages 292–305, 2000.
12. 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.
13. P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2005.
14. 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.
15. 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.
16. M. Frazier and L. Pitt. Learning from entailment: An application to propositional horn sentences. In *ICML*, pages 120–127, 1993.
17. M. Frazier and L. Pitt. Classic learning. *Machine Learning*, 25(2-3):151–193, 1996.
18. 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.
19. J. Lehmann and P. Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203–250, 2010.
20. C. Lutz, R. Piro, and F. Wolter. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *IJCAI*, pages 983–988, 2011.
21. L. D. Raedt. Logical settings for concept-learning. *Artif. Intell.*, 95(1):187–201, 1997.
22. C. Reddy and P. Tadepalli. Learning Horn definitions: Theory and an application to planning. *New Generation Comput.*, 17(1):77–98, 1999.
23. B. ten Cate, V. Dalmau, and P. G. Kolaitis. Learning schema mappings. In *ICDT*, pages 182–195, 2012.