

Exact Learning of Lightweight Description Logic Ontologies

Boris Konev

University of Liverpool
United Kingdom

Carsten Lutz

Universität Bremen
Germany

Ana Ozaki

University of Liverpool
United Kingdom

Frank Wolter

University of Liverpool
United Kingdom

Abstract

We study 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 DL-Lite with role inclusions and composite concepts on the right-hand side of concept inclusions can be learned in polynomial time; (2) $\mathcal{EL}$ TBoxes with only concept names on the right-hand side of concept inclusions can be learned in polynomial time; and (3) $\mathcal{EL}$ TBoxes cannot be learned in polynomial time. It follows that non-polynomial time learnability of $\mathcal{EL}$ TBoxes is caused by the interaction between existential restrictions on the right- and left-hand sides of concept inclusions. We also show that neither entailment nor equivalence queries alone are sufficient in cases (1) and (2) above.

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 (Baader et al. 2003), bootstrapping ontology design from examples, data, or text (Baader and Molitor 2000; Borchmann and Distel 2011; Ma and Distel 2013) and checking the completeness of the modeling in a systematic way (Baader et al. 2007).

Machine learning (ML) techniques are natural candidates to support 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. Some approaches also support learning of subsumptions between the identified terms, but not between compound logical expressions based on these terms (Cimiano, Hotho, and Staab 2005; Buitelaar, Cimiano, and Magnini 2005). In the context of DL ontologies, though, the main objective is to develop a detailed modeling of the relevant concepts of the domain using the logical operators provided by the DL at hand (Voelker 2009). In this paper, we thus concentrate on learning the full logical structure of a DL TBox. In particular, we study the foundations of learning ontologies that are formulated in several DLs which underly the three tractable *profiles* of the OWL2 ontology language.

We adopt Angluin et al.'s framework of exact learning via queries (Angluin 1987c). In particular, 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 who 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 $\mathcal{H}$ (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. While, in general, answering an equivalence query of the above form can be expected to be rather challenging also for a domain expert and an ontology engineer, we emphasize that the actual algorithms developed in this paper use equivalence queries only in a very restricted way. In particular, our algorithms always ensure that the hypothesis TBox $\mathcal{H}$ that they maintain is a logical consequence of the target TBox $\mathcal{T}$, and then use equivalence queries only to ask 'is the TBox $\mathcal{H}$ learned so far complete and if not, please give me a missing inclusion $C \sqsubseteq D$ '.

Following Angluin, 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, then $\mathcal{L}$ TBoxes are *polynomial time learnable*.

Within the setup laid out above, we study three different DLs, which are significant fragments of the three profiles of OWL2 (Motik et al. 2009). The popular description logic $\mathcal{EL}$ can be viewed as a logical core of the OWL2 EL profile of OWL2. 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 these types of queries is disallowed (Frazier and Pitt 1993; Angluin, Frazier, and Pitt 1992; Angluin 1987a), see also (Arias and Balcázar 2011). We show that, in contrast, $\mathcal{EL}$ TBoxes are not polynomial time learnable. The results for the other two DLs that we study will shed some light on the reasons for this behaviour.

The DL-Lite family of DLs underlies the OWL2 QL profile of OWL2 (Calvanese et al. 2007). In many versions of DL-Lite such as in DL-Lite $_{\mathcal{R}}$ (DL-Lite with role inclusions), the number of Σ -concept inclusions is polynomial in the size of Σ . In this case, TBoxes are trivially learnable in polynomial time, even when only entailment queries (but no equivalence queries) are available or vice versa. We thus consider the more interesting DL-Lite $_{\mathcal{R}}^{\exists}$ dialect, that is, the extension of DL-Lite $_{\mathcal{R}}$ that allows any $\mathcal{ELI}$ concept on the right-hand side of concept inclusions. Note that DL-Lite $_{\mathcal{R}}^{\exists}$ can be viewed as a logical core of OWL2 QL. Here, the number of Σ -concept inclusions is infinite, but we are nevertheless able to establish that DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes are polynomial time learnable. In practice, many $\mathcal{EL}$ TBoxes actually fall within (the intersection of $\mathcal{EL}$ and) DL-Lite $_{\mathcal{R}}^{\exists}$. We also show that DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes cannot be learned in polynomial time with entailment queries alone or with equivalence queries alone.

We then consider the fragment $\mathcal{EL}_{\text{lhs}}$ of $\mathcal{EL}$ where only concept names (but no compound concepts) are admitted on the right-hand side of concept inclusions. $\mathcal{EL}_{\text{lhs}}$ is a significant fragment of the OWL2 RL profile of OWL2, and also a fragment of datalog. We prove that $\mathcal{EL}_{\text{lhs}}$ TBoxes can be learned in polynomial time using a non-trivial extension of Angluin’s polynomial time algorithm for learning propositional Horn theories (1992). Note that the symmetric fragment $\mathcal{EL}_{\text{rhs}}$ that allows only concept names on the left-hand side of concept inclusions is a fragment of DL-Lite $_{\mathcal{R}}^{\exists}$. Together, our results for DL-Lite $_{\mathcal{R}}^{\exists}$ and $\mathcal{EL}_{\text{lhs}}$ thus show that non-polynomial time learnability of $\mathcal{EL}$ TBoxes is caused by the interaction between existential restrictions on the right- and left-hand sides of concept inclusions.

To improve readability, we first present the upper bounds

(polynomial time learnability for DL-Lite $_{\mathcal{R}}^{\exists}$ and $\mathcal{EL}_{\text{lhs}}$) and then the lower bounds. Proof details are provided in an appendix (Konev et al. 2013).

Related Work. In the terminology of the learning literature, entailment queries are a kind of membership queries, which can take many different forms (Raedt 1997). 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, polynomial time learnable classes for this protocol include regular sets (Angluin 1987b) and monotone DNF (Angluin 1987c). 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 (Reddy and Tadepalli 1999; Arias and Khardon 2002; Arias, Khardon, and Maloberti 2007; Selman and Fern 2011). Note that $\mathcal{EL}_{\text{lhs}}$ can be regarded as a fragment of first-order (FO) Horn logic. Existing approaches to polynomial time learning of FO Horn formulas 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. Also related to our work is exact learning of schema mappings in data exchange, as recently studied in (ten Cate, Dalmau, and Kolaitis 2012). 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 DL, exact learning has been studied for CLASSIC in (Frazier and Pitt 1996) where it is shown that single CLASSIC concepts (but not TBoxes) can be learned in polynomial time (here membership queries are concepts). Learning single concepts using refinement operators has been studied in (Lehmann and Hitzler 2010).

Preliminaries

Let N_C be a countably infinite set of *concept names* and N_R a countably infinite set of *role names*. The dialect DL-Lite $_{\mathcal{R}}^{\exists}$ of DL-Lite is defined as follows (Calvanese et al. 2007). A *role* is a role name or an inverse role r^- with $r \in N_R$. 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 $\exists r.T$, 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 and C is an $\mathcal{ELI}$ concept, 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 N_C and r ranges over N_R . A DL-Lite $_{\mathcal{R}}^{\exists}$ TBox is a finite set of DL-Lite $_{\mathcal{R}}^{\exists}$ CIs and RIs.¹

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 (Baader, Brandt, and Lutz 2005). 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 *acyclic* if it consists of inclusions $A \sqsubseteq C$ and

¹For simplicity, we consider DL-Lite without disjointness axioms. All our results also hold for the extension of DL-Lite $_{\mathcal{R}}^{\exists}$ with disjointness.

$A \equiv C$ 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 (Baader et al. 2003; Konev et al. 2012). The semantics of concepts and TBoxes is defined as usual in terms of interpretations (Baader et al. 2003). For a TBox $\mathcal{T}$ and concept inclusion $C \sqsubseteq D$, we write $\mathcal{T} \models C \sqsubseteq D$ if every model $\mathcal{I}$ of $\mathcal{T}$ also satisfies $C \sqsubseteq D$. $\mathcal{T}$ is omitted if it is empty, that is, we then simply write $\models C \sqsubseteq D$. We say that concepts C and D are *equivalent w.r.t. $\mathcal{T}$* and write $C \equiv_{\mathcal{T}} D$ if both $\mathcal{T} \models C \sqsubseteq D$ and $\mathcal{T} \models D \sqsubseteq C$; if r, s are roles, $r \equiv_{\mathcal{T}} s$ is defined analogously. 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 $\sum_{C \sqsubseteq D \in \mathcal{T}} |C| + |D|$.

Learning DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes in Polynomial Time

We prove that DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes can be learned in polynomial time. The signature Σ of the target TBox $\mathcal{T}$ is known to the learner. To simplify presentation, 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 $A_r \equiv \exists r. \top \in \mathcal{T}$. 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 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 it has learned.

In an initial phase, the learning algorithm for DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes determines, using at most $\mathcal{O}(|\Sigma|^2)$ entailment queries, all CIs $B_1 \sqsubseteq B_2$ with $\mathcal{T} \models B_1 \sqsubseteq B_2$ and B_1, B_2 basic concepts and the set of all RIs $r \sqsubseteq s$ with $\mathcal{T} \models r \sqsubseteq s$. A concept or role inclusion is in *reduced form* if all basic concepts that occur in it are concept names. We assume without further notice that all concept inclusions considered by the learner, except those that have been determined in the initial phase, are in reduced form. In particular, counterexamples returned by the oracle are immediately converted into this form.

To formulate 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. In detail, if A is a concept name, then T_A has a single node d with label $l(d) = \{A\}$; if $C = \exists r. D$, then T_C is obtained from T_D by adding a new root d_0 and an edge from d_0 to the root d of T_D with label $l(d_0, d) = r$ (we then call d an *r-successor* of d_0); if $C = D_1 \sqcap D_2$, then T_C is obtained by identifying the roots of T_{D_1} and T_{D_2} . 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 which allows us to speak, 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 $_{\mathcal{R}}^{\exists}$ TBox. A DL-Lite $_{\mathcal{R}}^{\exists}$

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

```

1: Compute (entailment queries)
    $\mathcal{H}_{basic} = \{r \sqsubseteq s \mid \mathcal{T} \models r \sqsubseteq s\} \cup$ 
    $\{B_1 \sqsubseteq B_2 \mid \mathcal{T} \models B_1 \sqsubseteq B_2, B_1, B_2 \text{ basic}\}$ 
2: Set  $\mathcal{H}_{add} = \emptyset$ 
3: while Equivalent  $(\mathcal{H}_{basic} \cup \mathcal{H}_{add})$ ? returns “no” do
4:   Let  $A \sqsubseteq C$  be the returned positive counterexample
     for  $\mathcal{T}$  relative to  $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ 
5:   Find a  $\mathcal{T}$ -essential 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
      $\mathcal{H}_{basic} \models C^* \sqsubseteq C'' \sqcap C'$  (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  $\mathcal{H} = \mathcal{H}_{basic} \cup \mathcal{H}_{add}$ .

```

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 *role saturated* for $\mathcal{T}$: if C' results from C by replacing a role r by a role r' with $r' \not\equiv_{\mathcal{T}} r$ and $\mathcal{T} \models r' \sqsubseteq r$, then $\mathcal{T} \not\models A \sqsubseteq C'$.
3. $A \sqsubseteq C$ is *sibling-merged* for $\mathcal{T}$: if C' is the result of identifying in C an r -successor and an s -successor of the same node where $r \equiv_{\mathcal{T}} s$, then $\mathcal{T} \not\models A \sqsubseteq C'$.²
4. $A \sqsubseteq C$ is *parent/child-merged* for $\mathcal{T}$: if d' is an r -successor of d in C and d'' is an s -successor of d' with $r \equiv_{\mathcal{T}} s$, then $\mathcal{T} \not\models A \sqsubseteq C'$ where C' results from C by identifying d and d'' .
5. $A \sqsubseteq C$ is *minimal* for $\mathcal{T}$: if d' is an r -successor of d in C and A' is in the node label of d , then $\mathcal{T} \not\models A' \sqsubseteq \exists r. C'$ where C' corresponds to the subtree rooted at d' (where $A' \not\equiv_{\mathcal{T}} A$ if d is the root of C).

Note that Points 1 to 5 carefully mix conditions that ‘maximize’ (Points 1 and 2) and ‘minimize’ (Point 5) the concept C . Points 3 and 4 are a form of maximization.

The algorithm for learning DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes in named form is given as Algorithm 1, where the implementation of the lines marked with “(entailment queries)” requires the execution of entailment queries as detailed below. Observe that, in Line 4, the assumption that a positive counterexample is returned by the oracle (i.e., CI entailed by $\mathcal{T}$ but not by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$) is justified by the construction of $\mathcal{H}_{basic}$ and $\mathcal{H}_{add}$, that is, we only include inclusions entailed by the target TBox $\mathcal{T}$ in our hypothesis TBox $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ and thus, at all times, $\mathcal{T} \models \mathcal{H}_{basic} \cup \mathcal{H}_{add}$. We now show in

²The node that results from the identification is then either an r -successor or an s -successor of its parent, which is equivalent.

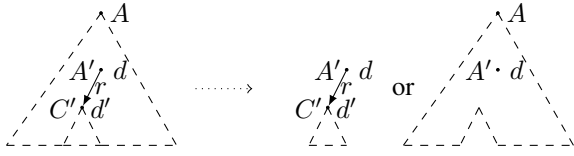


Figure 1: Minimization.

more detail how Lines 5 and 7 can be implemented, and that only polynomially many entailment queries are required. The next lemma addresses 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 such counterexample using only polynomially many entailment queries in $|C| + |\mathcal{T}|$.*

Proof. Let $A \sqsubseteq C$ be a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. 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:

(Concept 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 $A \sqsubseteq C'$.

(Role saturation) if $\mathcal{T} \models A \sqsubseteq C'$ and C' results from C by replacing a role r by a r' with $r \not\sqsubseteq_{\mathcal{T}} r'$ and $\mathcal{T} \models r' \sqsubseteq r$, 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 in C an r -successor and an s -successor of the same node where $r \equiv_{\mathcal{T}} s$, then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Parent/child merging) if d' is an r -successor of d in C , d'' is an s -successor of d' with $r \sqsubseteq_{\mathcal{T}} s$, and $\mathcal{T} \models A \sqsubseteq C'$ where C' results from C by identifying d and d'' , then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Minimization) if d' is an r -successor of d in C , A' is in the node label of d , and C' corresponds to the subtree rooted at d' , $\mathcal{T} \models A' \sqsubseteq \exists r.C'$ (plus $A' \not\sqsubseteq_{\mathcal{T}} A$ if d is the root of C), then replace $A \sqsubseteq C$ by

- (a) $A' \sqsubseteq \exists r.C'$ if $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A' \sqsubseteq \exists r.C'$; or
- (b) $A \sqsubseteq C|_{d'\downarrow}$, where $C|_{d'\downarrow}$ is obtained from C by removing the subtree generated by d' from C , otherwise.

The last rule is illustrated in Figure 1. It follows directly from the definition of the rules that the final concept inclusion is $\mathcal{T}$ -essential. Let us verify that it is also a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. It suffices to show that the CI resulting from each rule application is entailed by $\mathcal{T}$, but not by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. The former is straightforward for all five rules. Regarding the latter, in the first four rules we have $\mathcal{H}_{basic} \models C' \sqsubseteq C$ if $A \sqsubseteq C$ is replaced by $A \sqsubseteq C'$. Hence $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A \sqsubseteq C'$. In the last rule, we have $\{A \sqsubseteq C|_{d'\downarrow}, A' \sqsubseteq \exists r.C'\} \models A \sqsubseteq C$. Thus, one of the inclusions $A \sqsubseteq C|_{d'\downarrow}$ and $A' \sqsubseteq \exists r.C'$ is not entailed by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$, and this is the CI resulting from the rule application.

It is not hard to see that the number of rule applications is bounded by $|C|^2 \cdot |\Sigma|$. Note that termination of the algorithm

requires the condition in the last rule that $A' \not\sqsubseteq_{\mathcal{T}} A$ if d is the root of C . Otherwise, this rule could ‘replace’ $A \sqsubseteq C$ by any $A' \sqsubseteq C$ with A equivalent to A' w.r.t. $\mathcal{T}$. With the mentioned condition and since all CIs are in reduced form, each time when d is the root of C and $A \sqsubseteq C$ is replaced by $A' \sqsubseteq \exists r.C'$, then we strictly descend in subsumption order, that is, $\mathcal{T} \models A \sqsubseteq A'$ and $\mathcal{T} \not\models A' \sqsubseteq A$. $\square$

The following lemma addresses Line 7.

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 $\mathcal{H}_{basic} \models C \sqsubseteq C_1 \sqcap C_2$ using polynomially many entailment queries in $|C_1| + |C_2|$.*

Proof. Starting with $A \sqsubseteq C_1 \sqcap C_2$, we exhaustively apply the rule for sibling merging from Lemma 1 and denote the result by $A \sqsubseteq C$. Observe that since $A \sqsubseteq C_1$ and $A \sqsubseteq C_2$ are both $\mathcal{T}$ -essential, the concept inclusion $A \sqsubseteq C_1 \sqcap C_2$ is (i) concept-saturated for $\mathcal{T}$, (ii) role saturated for $\mathcal{T}$, (iii) parent/child-merged for $\mathcal{T}$, and (iv) minimal for $\mathcal{T}$. The only property of $\mathcal{T}$ -essential inclusions that can fail for $A \sqsubseteq C_1 \sqcap C_2$ is being sibling-merged for $\mathcal{T}$. Now one can show that an inclusion with properties (i)–(iv) still has those properties after applying the rule (sibling merging) above. Thus $A \sqsubseteq C$ is $\mathcal{T}$ -essential, as required. $\square$

Example 3 *Suppose the target TBox $\mathcal{T}$ is³*

Prof $\sqsubseteq$ Academic $\sqcap$
 $\exists \text{worksFor} . (\exists \text{supports} . \text{Teaching} \sqcap$
 $\exists \text{supports} . \text{Research})$
 Academic $\sqsubseteq \exists \text{advisor} . \text{Academic}$

The algorithm first computes $\mathcal{H}_{basic}$ as

Prof $\sqsubseteq$ Academic Prof $\sqsubseteq \exists \text{worksFor} . \top$
 Prof $\sqsubseteq \exists \text{advisor} . \top$ Academic $\sqsubseteq \exists \text{advisor} . \top$

and then poses $\mathcal{H}_{basic}$ as the first equivalence query. Assume that the oracle returns the positive counterexample

Prof $\sqsubseteq \exists \text{advisor} . \text{Academic}.$

which is not concept saturated for $\mathcal{T}$; saturating it yields

Prof $\sqsubseteq \text{Academic} \sqcap \exists \text{advisor} . \text{Academic}$

which is not minimal for $\mathcal{T}$; minimizing it yields

Academic $\sqsubseteq \exists \text{advisor} . \text{Academic}.$

The above CI constitutes the new $\mathcal{H}_{add}$ and the algorithm asks $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ as an equivalence query. Assume the oracle returns the positive counterexample

Prof $\sqsubseteq \exists \text{worksFor} . (\exists \text{supports} . \text{Teaching} \sqcap$
 $\exists \text{worksFor} . \text{Academic})$

³To facilitate presentation, we use a TBox $\mathcal{T}$ that is not in named form; for this concrete example, however, named form does not make any difference.

which is not parent/child-merged for $\mathcal{T}$; merging it results in the following CI, which is added to $\mathcal{H}_{\text{add}}$:

Prof $\sqsubseteq$ Academic $\sqcap \exists \text{worksFor}.\exists \text{supports}.\text{Teaching}.$

For the next equivalence query, the oracle returns the positive counterexample

Prof $\sqsubseteq \exists \text{worksFor}.\exists \text{supports}.\text{Research}.$

Following Lines 6 and 7 of Algorithm 1, we conjoin the right-hand side of this CI with that of the existing CI for Prof in $\mathcal{H}_{\text{add}}$. Applying sibling merging yields the first CI from $\mathcal{T}$, and the algorithm has succeeded to learn $\mathcal{T}$.

If Algorithm 1 terminates, then it obviously has found a TBox $\mathcal{H}_{\text{basic}} \cup \mathcal{H}_{\text{add}}$ that is logically equivalent to $\mathcal{T}$. It thus remains to show that the algorithm terminates in polynomial time. Observe that $\mathcal{H}_{\text{add}}$ contains at most one concept inclusion $A \sqsubseteq C$ for each concept name A . At each step in the while loop, either some $A \sqsubseteq C$ is added to $\mathcal{H}_{\text{add}}$ such that no inclusion with A on the left-hand side existed in $\mathcal{H}_{\text{add}}$ before or an existing inclusion $A \sqsubseteq C$ in $\mathcal{H}_{\text{add}}$ is replaced by a fresh $A \sqsubseteq C'$ with $\models C' \sqsubseteq C$ and $\not\models C \sqsubseteq C'$. It is thus not hard to see that we can prove termination in polynomial time by establishing the following lemma.

Lemma 4 *The number of replacements of an existing CI $A \sqsubseteq C$ in $\mathcal{H}_{\text{add}}$ is bounded polynomially in $|\mathcal{T}|$.*

Proof.(sketch) We first observe that, as a consequence of the fact that $A \sqsubseteq C$ and its replacement $A \sqsubseteq C'$ are both $\mathcal{T}$ -essential, the tree that corresponds to C is obtained from the tree that corresponds to C' by removing subtrees (and possibly replacing roles). Let n_C denote the number of nodes in the tree representation of C and let $A^{\mathcal{T}} = \{B \mid A \sqsubseteq B \in \mathcal{H}_{\text{basic}}\} \cup \{D \mid A \equiv_{\mathcal{T}} B, B \sqsubseteq D \in \mathcal{T}\}$. Then it suffices to prove the following

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

The claim is proved in the appendix using canonical models for DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes. $\square$

We have obtained the following main result of this section.

Theorem 5 *DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes are polynomial time learnable using entailment and equivalence queries.*

In the appendix, we provide additional examples showing that if any of the five conditions for being $\mathcal{T}$ -essential is removed, then Algorithm 1 no longer runs in polynomial time or does not even terminate. As an example, assume that minimality is omitted from the definition of being $\mathcal{T}$ -essential and let $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq \exists r.B\}$. Then the oracle can provide for the n -th equivalence query the positive counterexample $A \sqsubseteq \exists r^n.B$, $n \geq 1$. The algorithm does not terminate.

Learning $\mathcal{EL}_{\text{lhs}}$ TBoxes in Polynomial Time

We consider the restriction $\mathcal{EL}_{\text{lhs}}$ of general $\mathcal{EL}$ TBoxes where only concept names are allowed on the right-hand side of concept inclusions. We assume that CIs used in entailment queries and in equivalence queries and those returned as counterexamples are also of this restricted form. Our learning algorithm is a non-trivial extension of the polynomial time algorithm for learning propositional Horn theories presented in (Angluin, Frazier, and Pitt 1992; Arias and Balcázar 2011).

We first introduce some notation. An interpretation $\mathcal{I}$ is a *tree interpretation* if the directed graph $(\Delta^{\mathcal{I}}, \bigcup_{r \in \text{Nr}} r^{\mathcal{I}})$ is a tree and $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$ for all distinct $r, s \in \text{Nr}$. 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 (Lutz, Piro, and Wolter 2011):

Lemma 6 *For all $\mathcal{EL}$ concepts C : $d \in C^{\mathcal{I}}$ and $e \in C^{\mathcal{J}}$ iff $(d, e) \in C^{\mathcal{I} \times \mathcal{J}}$.*

One can show that the product of tree interpretations is a disjoint union of tree interpretations. If $\mathcal{I}$ and $\mathcal{J}$ are tree interpretations, we denote by $\mathcal{I} \times_r \mathcal{J}$ the tree interpretation that is contained in $\mathcal{I} \times \mathcal{J}$ with root $(\rho_{\mathcal{I}}, \rho_{\mathcal{J}})$.

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: (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 $(\mathcal{I}, d) \Rightarrow (\mathcal{J}, e)$ if there is a simulation from $(\mathcal{I}, d)$ to $(\mathcal{J}, e)$ and if $\mathcal{I}$ and $\mathcal{J}$ are tree interpretations then we write $\mathcal{I} \Rightarrow \mathcal{J}$ as a shorthand for $(\mathcal{I}, \rho_{\mathcal{I}}) \Rightarrow (\mathcal{J}, \rho_{\mathcal{J}})$. Simulations preserve the membership of $\mathcal{EL}$ -concepts (Lutz, Piro, and Wolter 2011):

Lemma 7 *For all $\mathcal{EL}$ concepts C : if $d \in C^{\mathcal{I}}$ and $(\mathcal{I}, d) \Rightarrow (\mathcal{J}, e)$, then $e \in C^{\mathcal{J}}$.*

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, that is, $\mathcal{I}_C$ is the tree T_C introduced in the context of our learning algorithm for DL-Lite $_{\mathcal{R}}^{\exists}$, represented as an interpretation. Conversely, every tree interpretation $\mathcal{I}$ can be viewed as an $\mathcal{EL}$ concept $C_{\mathcal{I}}$ in a straightforward way.

Let $\mathcal{T}$ be the target TBox to be learned, and assume again that its signature Σ is known to the learner. An interpretation $\mathcal{I}$ is a $\mathcal{T}$ -*countermodel* if $\mathcal{I} \not\models \mathcal{T}$. We will now describe a class of $\mathcal{T}$ -countermodels that are in a sense minimal and central to our learning algorithm. For a tree interpretation $\mathcal{I}$, we use $\mathcal{I}|_{\rho_{\mathcal{I}}}$ to denote the interpretation obtained from $\mathcal{I}$ by removing the root $\rho_{\mathcal{I}}$ of $\mathcal{I}$. For any element d of $\Delta^{\mathcal{I}}$, we use $\mathcal{I}|_{d\downarrow}$ to denote $\mathcal{I}$ with the subtree rooted at d removed. A $\mathcal{T}$ -countermodel is *essential* if the following conditions are satisfied:

Algorithm 2 The learning algorithm for $\mathcal{EL}_{\text{IHS}}$ TBoxes

```

1: Set  $\mathcal{J}$  to the empty sequence (of tree interpretations)
2: Set  $\mathcal{H} = \emptyset$ 
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  $\mathcal{J} \in \mathcal{J}$  such that  $\mathcal{J} \not\models (\mathcal{I} \times_r \mathcal{J})$  and
      $\mathcal{I} \times_r \mathcal{J} \not\models \mathcal{T}$  then
7:     Let  $\mathcal{J}$  be the first such element of  $\mathcal{J}$ 
8:     Find an essential  $\mathcal{T}$ -countermodel  $\mathcal{J}' \subseteq \mathcal{I} \times_r \mathcal{J}$ 
     (entailment queries)
9:     replace  $\mathcal{J}$  in  $\mathcal{J}$  with  $\mathcal{J}'$ 
10:  else
11:    append  $\mathcal{I}$  to  $\mathcal{J}$ 
12:  end if
13:  Construct  $\mathcal{H} = \{C_{\mathcal{I}} \sqsubseteq A \mid \mathcal{I} \in \mathcal{J}, \mathcal{T} \models C_{\mathcal{I}} \sqsubseteq A\}$ 
     (entailment queries)
14: end while

```

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

Intuitively, Condition 1 states that $\mathcal{I}$ contradicts $\mathcal{T}$ only at the root, that is, the only reason for why $\mathcal{I}$ does not satisfy $\mathcal{T}$ is that for at least one CI $C \sqsubseteq A \in \mathcal{T}$, we have that $\rho_{\mathcal{I}} \in C^{\mathcal{I}}$ and $\rho_{\mathcal{I}} \notin A^{\mathcal{I}}$. Condition 2 is a minimality condition which states that for any such $C \sqsubseteq A$ (which needs not be unique), $\rho_{\mathcal{I}}$ is no longer in $C^{\mathcal{I}}$ if we remove any node from $\mathcal{I}$.

The algorithm for learning $\mathcal{EL}_{\text{IHS}}$ TBoxes is given as Algorithm 2, where the implementation of the lines marked with “(entailment queries)” requires the execution of entailment queries as detailed below. It maintains a sequence $\mathcal{J}$ of tree interpretations that intuitively represents the TBox $\mathcal{H}$ constructed in Line 13; however, tree interpretations are easier to work with for our purposes than the $\mathcal{EL}$ concepts that they represent. In Line 8, we write $\mathcal{J}' \subseteq \mathcal{I} \times_r \mathcal{J}$ as shorthand for: $\mathcal{J}'$ is a subinterpretation of $\mathcal{I} \times_r \mathcal{J}$ that is obtained from $\mathcal{I} \times_r \mathcal{J}$ by removing subtrees. Note that the assumption in Line 4 that a *positive* counterexample is returned is justified by the construction of $\mathcal{H}$ in Lines 2 and 13, which ensures that, at all times, $\mathcal{T} \models \mathcal{H}$.

We now provide additional details on how to 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 $\mathcal{I} \in \mathcal{J}$ and A a concept name from Σ . We will later show that the length of $\mathcal{J}$ is bounded polynomially in $|\mathcal{T}|$, therefore polynomially many entailment queries suffice. Lines 5 and 8 are addressed by Lemmas 8 and 9 below.

Lemma 8 *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 $|\mathcal{T}| + |C|$.*

Proof. $\mathcal{I}$ is constructed by applying the following rules to $\mathcal{I} := \mathcal{I}_C$.

1. Saturate $\mathcal{I}$ by exhaustively applying the CIs from $\mathcal{H}$ as rules: if $D \sqsubseteq B \in \mathcal{H}$ and $d \in D^{\mathcal{I}}$, then add d to $B^{\mathcal{I}}$.
2. Replace $\mathcal{I}$ by a minimal subtree of $\mathcal{I}$ refuting $\mathcal{T}$ to address Condition 1 of essential $\mathcal{T}$ -countermodels: replace $\mathcal{I}$ by $\mathcal{I}|_d$ if $\mathcal{I}|_d$ is minimal with $\mathcal{I}|_d \not\models \mathcal{T}$ (checked using entailment queries), where $\mathcal{I}|_d$ denotes the subtree of $\mathcal{I}$ rooted at d .
3. Exhaustively remove subtrees from $\mathcal{I}$ until Condition 2 of essential $\mathcal{T}$ -countermodels is also satisfied: if $\mathcal{I}|_{d\downarrow} \not\models \mathcal{T}$ (checked using entailment queries), then replace $\mathcal{I}$ by $\mathcal{I}|_{d\downarrow}$.

The resulting interpretation $\mathcal{I}$ is as required. Details are presented in the appendix. $\square$

Lemma 9 *Given essential $\mathcal{T}$ -countermodels $\mathcal{I}$ and $\mathcal{J}$ with $\mathcal{I} \times_r \mathcal{J} \not\models \mathcal{T}$, one can construct an essential $\mathcal{T}$ -countermodel $\mathcal{J}' \subseteq \mathcal{I} \times_r \mathcal{J}$ using only polynomially many entailment queries in $|\mathcal{T}| + |\mathcal{I}| + |\mathcal{J}|$.*

Proof. Let $\mathcal{I}$ and $\mathcal{J}$ be essential $\mathcal{T}$ -countermodels with $\mathcal{I} \times_r \mathcal{J} \not\models \mathcal{T}$. Set $\mathcal{J}' = \mathcal{I} \times_r \mathcal{J}$ and then exhaustively apply Rule 3 from the proof of Lemma 8 to $\mathcal{J}'$. Clearly, the new $\mathcal{J}'$ is a $\mathcal{T}$ -countermodel. Moreover, it is essential:

(1) $\mathcal{J}'|_{\text{root}} \models \mathcal{T}$: by Lemma 6, $\mathcal{I}|_{\text{root}} \models \mathcal{T}$ and $\mathcal{J}|_{\text{root}} \models \mathcal{T}$ imply $\mathcal{I}|_{\text{root}} \times \mathcal{J}|_{\text{root}} \models \mathcal{T}$. Now $\mathcal{J}'|_{\text{root}}$ can be obtained from $\mathcal{I}|_{\text{root}} \times \mathcal{J}|_{\text{root}}$ by removing subtrees, and removing subtrees clearly preserves being a model of an $\mathcal{EL}_{\text{IHS}}$ TBox.

(2) $\mathcal{J}'|_{d\downarrow} \models \mathcal{T}$ for all $d \in \Delta^{\mathcal{J}'} \setminus \{\rho_{\mathcal{J}'}\}$: otherwise, the subtree rooted at d would have been removed during the construction of $\mathcal{J}'$. $\square$

Example 10 *Suppose that the target TBox is*

$$\begin{aligned} \exists \text{parent.Mouse} &\sqsubseteq \text{Mouse} \\ \exists \text{parent.Rabbit} &\sqsubseteq \text{Rabbit} \\ \exists \text{parent.T} &\sqsubseteq \text{Living} \end{aligned}$$

and assume that the oracle returns to the first equivalence query the positive counterexample $C \sqsubseteq \text{Living}$ with

$$C = \exists \text{parent.Mouse} \sqcap \exists \text{parent.Rabbit}$$

$\mathcal{I}_C$ is a $\mathcal{T}$ -countermodel that violates Condition 2 of being essential which leads to its rewriting into $\mathcal{I}_{C'}$ with

$$C' = \exists \text{parent.Mouse}$$

(Alternatively, it could also lead to $C' = \exists \text{parent.Rabbit}$.) The algorithm inserts $\mathcal{I}_{C'}$ into $\mathcal{J}$ and asks the next equivalence query, to which the oracle returns $D \sqsubseteq \text{Living}$ with

$$D = \exists \text{parent.Rabbit}.$$

$\mathcal{I}_D$ is an essential $\mathcal{T}$ -countermodel and we have $\mathcal{I}_{C'} \not\models (\mathcal{I}_D \times_r \mathcal{I}_{C'})$, where $\mathcal{I}_D \times_r \mathcal{I}_{C'}$ corresponds to the concept

$$D' = \exists \text{parent.T}.$$

$\mathcal{I}_{D'}$ is $\mathcal{T}$ -essential and thus it is inserted into $\mathfrak{J}$ in place of $\mathcal{I}_{C'}$. At this point, the algorithm has learnt $\mathcal{H} = \{\exists \text{parent}.\top \sqsubseteq \text{Living}\}$. The algorithm asks the next equivalence query, to which the oracle returns $E \sqsubseteq \text{Rabbit}$ with

$$E = \exists \text{parent}.\exists \text{parent}.\text{Rabbit}$$

$\mathcal{I}_E$ violates Conditions 1 and 2 of being essential, which leads to its rewriting into $\mathcal{I}_{E'}$ with

$$E' = \text{Living} \sqcap \exists \text{parent}.\text{Rabbit}.$$

We have $\mathcal{I}_{D'} \Rightarrow \mathcal{I}_{E'} \times_r \mathcal{I}_{D'}$ and so $\mathcal{I}_{E'}$ is appended to the list $\mathfrak{J}$. At this point, the algorithm has learnt $\mathcal{H} = \{\text{Living} \sqcap \exists \text{parent}.\text{Rabbit} \sqsubseteq \text{Rabbit}, \exists \text{parent}.\top \sqsubseteq \text{Living}\}$, which is equivalent to the TBox that consists of the second and third CI in $\mathcal{T}$. The algorithm can now proceed to learn $\text{Living} \sqcap \exists \text{parent}.\text{Mouse} \sqsubseteq \text{Mouse}$.

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

Lemma 11 *Let $\mathfrak{J}$ be a sequence computed at some point of an execution of Algorithm 2. Then (i) the length of $\mathfrak{J}$ is bounded by the number of CIs in $\mathcal{T}$ and (ii) each interpretation in each position of $\mathfrak{J}$ is replaced only $|\mathcal{T}| + |\mathcal{T}|^2$ often with a new interpretation.*

Proof.(sketch) Assume that at each point of the execution of the algorithm, $\mathfrak{J}$ has the form $\mathcal{I}_0, \dots, \mathcal{I}_k$ for some $k \geq 0$. To establish Point (i), we closely follow (Angluin, Frazier, and Pitt 1992) and show that (iii) for every $\mathcal{I}_i$, there is a $D_i \sqsubseteq A_i \in \mathcal{T}$ with $\mathcal{I}_i \not\models D_i \sqsubseteq A_i$ and (iv) if $i \neq j$, then $D_i \sqsubseteq A_i$ and $D_j \sqsubseteq A_j$ are not identical.

The proof of Points (iii) and (iv), in turn requires a careful analysis of the concept inclusions in the target TBox that are violated at the root of the $\mathcal{T}$ -countermodels $\mathcal{I}_0, \dots, \mathcal{I}_k$.

For the proof of Point (ii), we show that $|\Delta^{\mathcal{I}}| \leq |\mathcal{T}|$ for any essential $\mathcal{T}$ -countermodel $\mathcal{I}$ and analyze the relationship between the essential $\mathcal{T}$ -countermodels produced from $\mathcal{I} \times_r \mathcal{J}$ when $\mathcal{J} \not\models (\mathcal{I} \times_r \mathcal{J})$ (using Lemma 6 and 7). $\square$

We have thus established the main result of this section.

Theorem 12 *$\mathcal{EL}_{\text{lhs}}$ TBoxes are polynomial time learnable using entailment and equivalence queries.*

It can again be shown that all adopted conditions are indeed necessary. Consider for example the first condition of being $\mathcal{T}$ -essential. Without it, for the target TBox $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$ the oracle can return the infinite set of positive counterexamples $\exists r^n.A \sqsubseteq A$, n a prime number. Then the learning algorithm does not terminate.

Limits of Polynomial Time Learnability

The main result of this section is that $\mathcal{EL}$ TBoxes cannot be learned in polynomial time using entailment and equivalence queries. We also show that $\text{DL-Lite}_{\mathcal{R}}^{\exists}$ TBoxes cannot be learned in polynomial time using entailment or

equivalence queries alone. This holds for $\mathcal{EL}_{\text{lhs}}$ TBoxes as well and follows from the fact that propositional Horn theories cannot be learned in polynomial time using entailment or equivalence queries alone (Frazier and Pitt 1993; Angluin, Frazier, and Pitt 1992; Angluin 1987a).

We start by proving the non-learnability result for $\mathcal{EL}$ TBoxes. Our proof shows that even *acyclic* target TBoxes cannot be learned in polynomial time when general $\mathcal{EL}$ concept inclusions are admitted in entailment queries, as counterexamples returned by the oracle, and in TBoxes used as equivalence queries. On our way, we also prove non-learnability of $\text{DL-Lite}_{\mathcal{R}}^{\exists}$ TBoxes using entailment queries only. The proof is inspired by Angluin's lower bound for the following abstract learning problem (1987c): a learner aims to identify one of N distinct sets $L_1, \dots, L_N$ which have the property that there exists a set $L_{\cap}$ for which $L_i \cap L_j = L_{\cap}$, for any $i \neq j$. It is assumed that $L_{\cap}$ is not a valid argument to an equivalence query. The learner can pose membership queries " $x \in L?$ " and equivalence queries " $H = L?$ ". Then in the worst case it takes at least $N - 1$ membership and equivalence queries to exactly identify a hypothesis L_i from $L_1, \dots, L_N$. The proof proceeds as follows. At every stage of computation, the oracle (which here should be viewed as an adversary) maintains a set of hypotheses S , which the learner is not able to distinguish based on the answers given so far. Initially, $S = \{L_1, \dots, L_N\}$. When the learner asks a membership query x , the oracle returns 'Yes' if $x \in L_{\cap}$ and 'No' otherwise. In the latter case, the (unique) L_i such that $x \in L_i$ is removed from S . When the learner asks an equivalence query H , the oracle returns 'No' and a counterexample $x \in L_{\cap} \oplus H$ (the symmetric difference of $L_{\cap}$ and H). This always exists as $L_{\cap}$ is not a valid query. If the counterexample x is not a member of $L_{\cap}$, (at most one) $L_i \in S$ such that $x \in L_i$ is eliminated from S . In the worst case, the learner has to reduce the cardinality of S to one to exactly identify a hypothesis, which takes $N - 1$ queries.

Similarly to the method outlined above, in our proof we maintain a set of acyclic $\mathcal{EL}$ TBoxes 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 size 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 in S , fix two role names r and s . For any sequence $\sigma = \sigma^1 \sigma^2 \dots \sigma^n$ with $\sigma^i \in \{r, s\}$, the expression $\exists \sigma.C$ stands for $\exists \sigma^1. \exists \sigma^2 \dots \exists \sigma^n.C$. For every such sequence σ , of which there are $N = 2^n$ many, consider the acyclic $\mathcal{EL}$ TBox $\mathcal{T}_{\sigma}$ defined as

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

where $\mathcal{T}_0$ generates a full binary tree whose edges are la-

belled with the role names r and s and with X_0 at the root, X_1 at level 1 and so on. M is a concept name that ‘marks’ a particular path in this tree given by the sequence σ . One can use Angluin’s strategy to show that TBoxes from the set S of all such TBoxes $\mathcal{T}_\sigma$ cannot be learned in polynomial time using entailment queries only: notice that for no sequence $\sigma' \neq \sigma$ of length n , we have $\mathcal{T}_\sigma \models A \sqsubseteq \exists \sigma'. M$. Thus an entailment query of the form $A \sqsubseteq \exists \sigma. M$ eliminates at most one TBox from the set of TBoxes that the learner cannot distinguish. This observation can be generalized to arbitrary entailment queries $C \sqsubseteq D$ in $\mathcal{EL}$ since one can prove, similarly to the proof of Lemma 14 below, that for any $\mathcal{EL}$ concept inclusion $C \sqsubseteq D$ either $\{A \sqsubseteq X_0\} \cup \mathcal{T}_0 \models C \sqsubseteq D$ (thus $C \sqsubseteq D$ is entailed by all TBoxes in S) 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. Observe that the TBoxes $\mathcal{T}_\sigma$ are actually formulated in $\text{DL-Lite}_{\mathcal{R}}^{\exists}$. We show in the appendix that all arguments above are true also when entailment queries are formulated in $\text{DL-Lite}_{\mathcal{R}}^{\exists}$ (which include inverse roles) and thus obtain the following result.

Theorem 13 *DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes are not polynomial time learnable using entailment queries, only.*

Interestingly, 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 $\mathcal{T}_\sigma$ as $A \sqsubseteq \exists \sigma. M$ can be found ‘inside’ every counterexample. Our strategy to rule out this option for the oracle is to modify $\mathcal{T}_1, \dots, \mathcal{T}_N$ in such a way that although a TBox $\mathcal{T}_i$ axiomatizing the intersection over the set of consequences of each $\mathcal{T}_i$, $i \leq N$, exists, its size is superpolynomial and so cannot be used as an equivalence query by a polynomial time learning algorithm.

For every $n > 0$ and every n -tuple $L = (\sigma_1, \dots, \sigma_n)$, where every σ_i is a role sequence of length n as above, we define an acyclic $\mathcal{EL}$ TBox $\mathcal{T}_L$ as the union of $\mathcal{T}_0$ and the following inclusions:⁴

$$\begin{aligned} A_1 &\sqsubseteq \exists \sigma_1. M \sqcap X_0 & \dots & A_n &\sqsubseteq \exists \sigma_n. M \sqcap X_0 \\ B_1 &\sqsubseteq \exists \sigma_1. M \sqcap X_0 & \dots & B_n &\sqsubseteq \exists \sigma_n. M \sqcap X_0 \\ A &\equiv X_0 \sqcap \exists \sigma_1. M \sqcap \dots \sqcap \exists \sigma_n. M. \end{aligned}$$

Let $\mathcal{L}_n$ be a set of n -tuples such that for $1 \leq i \leq n$ and every $L, L' \in \mathcal{L}_n$ with $L = (\sigma_1, \dots, \sigma_n)$, $L' = (\sigma'_1, \dots, \sigma'_n)$, if $\sigma_i = \sigma'_j$ then $L = L'$ and $i = j$. Then for any sequence σ of length n there exists at most one $L \in \mathcal{L}_n$ and at most one $i \leq n$ such that $\mathcal{T}_L \models A_i \sqsubseteq \exists \sigma. M$ and $\mathcal{T}_L \models B_i \sqsubseteq \exists \sigma. M$. We can choose $\mathcal{L}_n$ such that there are $N = \lfloor 2^n/n \rfloor$ different tuples in $\mathcal{L}_n$. Notice that the size of each $\mathcal{T}_L$ with $L \in \mathcal{L}_n$ is polynomial in n and so N is superpolynomial in the size of each $\mathcal{T}_L$ with $L \in \mathcal{L}_n$.

Every $\mathcal{T}_L$, for $L \in \mathcal{L}_n$, entails, among other inclusions, $\bigcap_{i=1}^n C_i \sqsubseteq A$, where every C_i is either A_i or B_i . There are

⁴In fact, to prove non-polynomial learnability, it suffices to consider $\exists \sigma_1. M \sqcap \dots \sqcap \exists \sigma_n. M \sqsubseteq A$ in place of the concept equality; however, inclusions of this form are not allowed in acyclic TBoxes.

2^n different such inclusions, which indicates that every representation of the ‘intersection TBox’ requires superpolynomially many axioms. It follows from Lemma 15 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. We use Σ_n to denote the signature of $\mathcal{T}_L$.

Lemma 14 *For all $\mathcal{EL}$ concept inclusions $C \sqsubseteq D$ over Σ_n :*

- either $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathcal{L}_n$ or
- the number of different $L \in \mathcal{L}_n$ 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 $L = (\sigma_1, \dots, \sigma_n)$ and $L' = (\sigma'_1, \dots, \sigma'_n)$. Then the inclusion $X_0 \sqcap \exists \sigma_1. M \sqcap \exists \sigma'_1. M \sqcap A_2 \sqcap \dots \sqcap A_n \sqsubseteq A$ is entailed by both $\mathcal{T}_L$ and $\mathcal{T}_{L'}$ but not by any other $\mathcal{T}_{L''}$ with $L \in \mathcal{L}_n$.

We now show how the oracle can answer equivalence queries, aiming to show that for any polynomial size equivalence query $\mathcal{H}$, the oracle can return a counterexample $C \sqsubseteq D$ such that either (i) $\mathcal{H} \models C \sqsubseteq D$ and $\mathcal{T}_L \models C \sqsubseteq D$ for at most one $L \in \mathcal{L}_n$ or (ii) $\mathcal{H} \not\models C \sqsubseteq D$ and for every $L \in \mathcal{L}_n$ 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 that the learner cannot distinguish. In addition, however, we have to take extra care of the size of counterexamples as the learning algorithm is allowed to run in time polynomial not only in the size of the target TBox but also in the size of the counterexamples returned by the oracle. For instance, if the hypothesis TBox $\mathcal{H}$ contains an inclusion $C \sqsubseteq D$ which is not entailed by any $\mathcal{T}_L$, one cannot simply return $C \sqsubseteq D$ as a counterexample since the learner will be able to ‘pump up’ its running time by asking for equivalence of the target TBox a sequence of hypotheses $\mathcal{H}_i = \{C_i \sqsubseteq D_i\}$ such that the size of $C_{i+1} \sqsubseteq D_{i+1}$ is twice the size of $C_i \sqsubseteq D_i$. Then at every stage in a run of the learning algorithm, the 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 15 *For any $n > 1$ and any $\mathcal{EL}$ TBox $\mathcal{H}$ in Σ_n with $|\mathcal{H}| < 2^n$, there exists an $\mathcal{EL}$ CI $C \sqsubseteq D$ over Σ_n such that (i) the size of $C \sqsubseteq D$ does not exceed $6n$ and (ii) if $\mathcal{H} \models C \sqsubseteq D$ then $\mathcal{T}_L \models C \sqsubseteq D$ for at most one $L \in \mathcal{L}_n$ and if $\mathcal{H} \not\models C \sqsubseteq D$ then for every $L \in \mathcal{L}_n$ we have $\mathcal{T}_L \models C \sqsubseteq D$.*

Then we have the following.

Theorem 16 *$\mathcal{EL}$ TBoxes are not polynomial time learnable using entailment and equivalence queries.*

Proof. Assume that TBoxes are polynomial time learnable. Then there exists a learning algorithm whose running time is bounded at any stage by a polynomial $p(n, m)$. Choose n such that $\lfloor 2^n/n \rfloor > (p(n, 6n))^2$ and let $S = \{\mathcal{T}_L \mid L \in \mathcal{L}_n\}$. We follow Angluin's strategy of letting the oracle remove 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 $L \in \mathcal{L}_n$, then the answer is 'yes'; otherwise the answer is 'no' and all $\mathcal{T}_L$ with $\mathcal{T}_L \models C \sqsubseteq D$ are removed from S (by Lemma 14, there are at most $|C|$ such TBoxes). Given an equivalence query $\mathcal{H}$, the answer is 'no', a counterexample $C \sqsubseteq D$ guaranteed by Lemma 15 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 no more than $p(n, 6n)$ queries and the size of every query does not exceed $p(n, 6n)$. By Lemmas 14 and 15, 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 and we have derived a contradiction. $\square$

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

Theorem 17 (Angluin) *For any polynomial $q(\cdot)$ there exist constants t_0 and s_0 and a strategy for the oracle $\mathfrak{D}$ to answer equivalence queries posed by a learning algorithm in such a way that for sufficiently large n any learning algorithm that asks at most $q(n)$ equivalence queries, each bounded in size by $q(n)$, cannot exactly identify elements of $M(n, t_0, s_0)$.*

To employ Theorem 17, we associate with every monotone DNF formula $\phi = \bigvee_{i=1}^t (x_1^i \wedge \dots \wedge x_{s_i}^i)$ with variables $x_1, \dots, x_n$ a DL-Lite $_{\mathcal{R}}^{\exists}$ TBox $\mathcal{T}_\phi$ as follows. With each conjunct $x_1^i \wedge \dots \wedge x_{s_i}^i$ we associate a concept $C_i := \exists \rho_1^i. \exists \rho_2^i. \dots \exists \rho_n^i. \top$ where $\rho_j^i = r$ if x_j occurs in $x_1^i \wedge \dots \wedge x_{s_i}^i$ and $\rho_j^i = \bar{r}$ otherwise (r and $\bar{r}$ are role names). Let A be a concept name and set

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

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

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

A truth assignment I (for the variables $x_1, \dots, x_n$) also corresponds to a concept $C_I := \exists \rho_1^i. \exists \rho_2^i. \dots \exists \rho_n^i. \top$, where $\rho_j^i = r$ if I makes x_j true and $\rho_j^i = \bar{r}$ otherwise. Then $I \models \phi$ iff $\mathcal{T}_\phi \models A \sqsubseteq C_I$ holds for all truth assignments I .

Note that $\bar{r}$ represents that a variable is false and r that a variable is true. Thus, the role inclusion $\bar{r} \sqsubseteq r$ captures the monotonicity of the DNF formulas considered. For any fixed values n, s and t , we set $T(n, t, s) = \{\mathcal{T}_\phi \mid \phi \in M(n, t, s)\}$.

Theorem 18 *The class of DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes is not polynomial time learnable using equivalence queries.*

Proof. We sketch the proof for the case when the TBoxes $\mathcal{H}$ from any equivalence query represent a monotone DNF formula in the variables $x_1, \dots, x_n$, that is, if all equivalence queries $\mathcal{H}$ are of the form

$$\{A \sqsubseteq \bigcap_{\rho_1 \dots \rho_n \in \Gamma} \exists \rho_1. \exists \rho_2. \dots \exists \rho_n. \top, \quad \bar{r} \sqsubseteq r\},$$

for some $\Gamma \subseteq \{r, \bar{r}\}^n$. Arbitrary DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes in equivalence queries are considered in the appendix.

For a proof by contradiction, suppose that the running time of a learning algorithm $\mathfrak{A}$ for DL-Lite $_{\mathcal{R}}^{\exists}$ TBoxes in $\Sigma = \{A, r, \bar{r}\}$ is bounded at every stage of computation by a polynomial $p(x, y)$, where x is the size of the target TBox, and y is the maximal size of a counterexample returned by the oracle up to the current stage of computation. Let $q(n) = p(n^2, 4n + 2)$, and let t_0 and s_0 be the constants from the strategy of the oracle in Theorem 17. Let n be sufficiently large so that the claim of Theorem 17 holds.

Consider the oracle $\mathfrak{D}'$ that responds to every equivalence query $\mathcal{H}$ that represents a monotone DNF-formula ψ in n variables by returning the counterexample $A \sqsubseteq C_I$ corresponding to the truth assignment I that $\mathfrak{D}$ would return for the equivalence query ψ . We show that $\mathfrak{A}$ cannot distinguish between certain TBoxes in $T(n, t_0, s_0)$ and thus obtain a contradiction.

The largest counterexample returned by $\mathfrak{D}'$ is of the form $A \sqsubseteq \exists \rho_1. \dots \exists \rho_n. \top$, so for sufficiently large n the maximal size of any counterexample returned by $\mathfrak{D}'$ in any run of the algorithm is bounded by $4n + 2$. Similarly, the size of every potential target TBox $\mathcal{T}_\phi \in T(n, t_0, s_0)$ does not exceed $t_0 \cdot (4n + 2)$ and, as t_0 is a constant, for sufficiently large n it is bounded by n^2 . Thus, for sufficiently large n the total running time of $\mathfrak{A}$ for any target TBox in $T(n, t_0, s_0)$ is bounded by $p(n^2, 4n + 2)$. So, the size of a monotone DNF equivalence query forwarded to the strategy $\mathfrak{D}$ is bounded by $q(n)$, and there will be at most $q(n)$ queries forwarded. But then $\mathfrak{D}$ returns answers such that some ϕ and ψ from $M(n, t_0, s_0)$ are not distinguished. It remains to observe that $\mathfrak{A}$ does not distinguish $\mathcal{T}_\phi$ and $\mathcal{T}_\psi$. $\square$

Future Work

We have presented the first study of learnability of DL TBoxes in Angluin et al's framework of learning via queries. Many research questions remain to be explored. An immediate question is whether acyclic $\mathcal{EL}$ TBoxes can be learned in polynomial time using queries and counterexamples of the form $A \equiv C$ and $A \sqsubseteq C$ only. Note that our non-learnability result for acyclic $\mathcal{EL}$ TBoxes relies heavily on counterexamples that are not of this form. Another immediate question is whether the extension of $\mathcal{EL}_{\text{lhs}}$ with inverse roles (which is a better approximation of OWL2 RL than $\mathcal{EL}_{\text{lhs}}$ itself) can still

be learned in polynomial time. Other interesting research directions are non-polynomial time learning algorithms for $\mathcal{EL}$ TBoxes and the admission of different types of membership queries and counterexamples in the learning protocol. For example, one could replace concept inclusions as counterexamples with interpretations. In an OBDA context, one could allow membership queries that speak about certain answers to queries over an ABox and relative to the target TBox. Our results provide a good starting point for studying such variations.

Acknowledgements Konev and Wolter were supported by the EPSRC project EP/H043594/1. Lutz was supported by the DFG project Prob-DL (LU1417/1-1). Ozaki is supported by the Science without Borders scholarship programme.

References

- Angluin, D.; Frazier, M.; and Pitt, L. 1992. Learning conjunctions of Horn clauses. *Machine Learning* 9:147–164.
- Angluin, D. 1987a. Learning propositional Horn sentences with hints. Technical report, Yale University.
- Angluin, D. 1987b. Learning regular sets from queries and counterexamples. *Inf. Comput.* 75(2):87–106.
- Angluin, D. 1987c. Queries and concept learning. *Machine Learning* 2(4):319–342.
- Angluin, D. 1990. Negative results for equivalence queries. *Machine Learning* 5:121–150.
- Arias, M., and Balcázar, J. L. 2011. Construction and learnability of canonical Horn formulas. *Machine Learning* 85(3):273–297.
- Arias, M., and Khardon, R. 2002. Learning closed Horn expressions. *Inf. Comput.* 178(1):214–240.
- Arias, M.; Khardon, R.; and Maloberti, J. 2007. Learning Horn expressions with LOGAN-H. *Journal of Machine Learning Research* 8:549–587.
- Baader, F., and Molitor, R. 2000. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In *ICCS*, 292–305.
- Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. 2003. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.
- Baader, F.; Ganter, B.; Sertkaya, B.; and Sattler, U. 2007. Completing description logic knowledge bases using formal concept analysis. In *IJCAI*, 230–235.
- Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, 364–369. Professional Book Center.
- Borchmann, D., and Distel, F. 2011. Mining of $\mathcal{EL}$ -GCIs. In *The 11th IEEE International Conference on Data Mining Workshops*. Vancouver, Canada: IEEE Computer Society.
- Buitelaar, P.; Cimiano, P.; and Magnini, B., eds. 2005. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press.
- Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated reasoning* 39(3):385–429.
- Cimiano, P.; Hotho, A.; and Staab, S. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res. (JAIR)* 24:305–339.
- Frazier, M., and Pitt, L. 1993. Learning from entailment: An application to propositional horn sentences. In *ICML*, 120–127.
- Frazier, M., and Pitt, L. 1996. Classic learning. *Machine Learning* 25(2-3):151–193.
- Konev, B.; Ludwig, M.; Walther, D.; and Wolter, F. 2012. The logical difference for the lightweight description logic *EL*. *J. Artif. Intell. Res. (JAIR)* 44:633–708.
- Konev, B.; Lutz, C.; Ozaki, A.; and Wolter, F. 2013. Appendix of: Exact learning of lightweight description logic ontologies. <http://cgi.csc.liv.ac.uk/~frank/publ/publ.html>.
- Lehmann, J., and Hitzler, P. 2010. Concept learning in description logics using refinement operators. *Machine Learning* 78(1-2):203–250.
- Lutz, C.; Piro, R.; and Wolter, F. 2011. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *IJCAI*, 983–988.
- Ma, Y., and Distel, F. 2013. Learning formal definitions for *snomed ct* from text. In *AIME*, 73–77.
- Motik, B.; Grau, B. C.; Horrocks, I.; Wu, Z.; Fokoue, A.; and Lutz, C. 2009. OWL 2 web ontology language: Profiles. W3C Recommendation.
- Raedt, L. D. 1997. Logical settings for concept-learning. *Artif. Intell.* 95(1):187–201.
- Reddy, C., and Tadepalli, P. 1999. Learning Horn definitions: Theory and an application to planning. *New Generation Comput.* 17(1):77–98.
- Selman, J., and Fern, A. 2011. Learning first-order definite theories via object-based queries. In *ECML/PKDD (3)*, 159–174.
- ten Cate, B.; Dalmau, V.; and Kolaitis, P. G. 2012. Learning schema mappings. In *ICDT*, 182–195.
- Voelker, J. 2009. *Learning Expressive Ontologies*. Ph.D. Dissertation, Universitaet Karlsruhe.