

# Exact Learning of TBoxes in EL and DL-Lite

Boris Konev<sup>1</sup> and Carsten Lutz<sup>2</sup> and Frank Wolter<sup>1</sup>

<sup>1</sup> Department of Computer Science, University of Liverpool, UK

<sup>2</sup> 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<sub>core</sub> with  $\mathcal{EL}\mathcal{T}$ -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 relevant 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  $\Sigma$  (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  $\Sigma$ -concept inclusion  $C \sqsubseteq D$ ? These queries are answered by the oracle with ‘yes’ or ‘no’.
- *equivalence queries*: is a given  $\Sigma$ -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  $\Sigma$ -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  $\Sigma$ , the number of  $\mathcal{L}$ -concept inclusions is polynomial in the size of  $\Sigma$ , 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<sub>core</sub>-TBoxes and DL-Lite<sub>R</sub>-TBoxes are polynomial time learnable. We thus start our investigation with considering the more interesting case of DL-Lite<sub>∃</sub>, which is the extension of DL-Lite<sub>core</sub> that allows any  $\mathcal{ELI}$ -concept on the right-hand side of concept inclusions. Although the number of  $\Sigma$ -concept inclusions is no longer polynomial in the size of  $\Sigma$ , we are still able to establish that DL-Lite<sub>∃</sub>-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  $\mathcal{EL}_{\text{lhs}}$  of  $\mathcal{EL}$  in which TBoxes consist of concept inclusions with only concept names on the right hand side. It turns out that  $\mathcal{EL}_{\text{lhs}}$  TBoxes are polynomial time learnable. Note that the corresponding fragment  $\mathcal{EL}_{\text{rhs}}$  that allows only concept names on the left hand side of concept inclusions is a fragment of DL-Lite<sub>∃</sub>, 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 [22]. 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 [23, 6]. Note that  $\mathcal{EL}_{\text{lhs}}$  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 [24]. 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 [20].

## 2 Preliminaries

Let  $\mathbb{N}_C$  be a countably infinite set of *concept names* and  $\mathbb{N}_R$  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  $r \in \mathbb{N}_R$ . 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  $\mathcal{EL}$ -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  $\mathbb{N}_C$  and  $r$  ranges over  $\mathbb{N}_R$ . A *DL-Lite $_{\exists}$  TBox* is a finite set of DL-Lite $_{\exists}$  inclusions. As usual, an  $\mathcal{EL}$ -*concept* is an  $\mathcal{EL}$ -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*  $\Sigma$  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<sub>∃</sub>-TBoxes are Learnable in Polynomial Time

We prove that DL-Lite<sub>∃</sub>-TBoxes can be learned in polynomial time.<sup>3</sup> 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  $\Sigma$ ), and replaces each  $A_r$  with  $\exists r.\top$  in queries to the oracle and when ultimately returning the TBox that it has learned.

As a first step, the learning algorithm for DL-Lite<sub>∃</sub> 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  $\Sigma$  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<sub>∃</sub> 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  $(a_{D \cap \exists r.D'}, a_{D'})$  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<sub>∃</sub> TBox. A DL-Lite<sub>∃</sub> 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  $C$ ).

<sup>3</sup> 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<sub>∃</sub>. It is also conceivable to use more expressive logics instead.

---

**Algorithm 1** The learning algorithm for DL-Lite<sub>∃</sub>

---

```
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<sub>∃</sub> TBoxes is given as Algorithm 1. In the remainder of this section, we prove that this algorithm is indeed capable of learning DL-Lite<sub>∃</sub> 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<sub>∃</sub> 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.)  $\square$

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

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

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  $\sigma$ . 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: notice that for no sequence  $\sigma' \neq \sigma$  of length  $n$  we have  $\mathcal{T}_\sigma \models A \sqsubseteq \exists \sigma'.M$ . Thus, following Angluin’s strategy, 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 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  $\mathcal{T}_\sigma$  as  $A \sqsubseteq \exists\sigma.M$  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  $L = (\sigma_1, \dots, \sigma_n)$ , where every  $\sigma_i$  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:<sup>4</sup>

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

$\Sigma_n$  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  $L = (\sigma_1, \dots, \sigma_n)$ ,  $L' = (\sigma'_1, \dots, \sigma'_n)$ , if  $\sigma_i = \sigma'_j$  then  $L = L'$  and  $i = j$ . Then for any sequence  $\sigma$  of length  $n$  there exists a unique  $L \in \mathfrak{L}_n$  and a unique  $i \leq n$  such that  $\mathcal{T}_L \models A_i \sqsubseteq \exists\sigma.M$  and  $\mathcal{T}_L \models B_i \sqsubseteq \exists\sigma.M$ . 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,  $\prod_{i=1}^n C_i \sqsubseteq A$ , 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  $\Sigma_n$ :*

- 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  $L = (\sigma_1, \dots, \sigma_n)$  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  $\mathcal{T}_{L''}$ .

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.

<sup>4</sup> In 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.

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  $\Sigma_n$  with  $|\mathcal{H}| < 2^n$  there exists an  $\mathcal{EL}$  CI  $C \sqsubseteq D$  over  $\Sigma_n$  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  $\mathcal{T}_L \not\models C \sqsubseteq D$ .*

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.  $\square$

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

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. 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 algorithm for learning propositional Horn theories from [4].

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

---

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

---

```
1: Let  $\mathfrak{J}$  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  $\mathcal{J} \in \mathfrak{J}$  such that  $\mathcal{J} \not\approx (\mathcal{I} \times_r \mathcal{J})$  and  $\mathcal{I} \times_r \mathcal{J} \not\models \mathcal{T}$  then
7:     Let  $\mathcal{J}'$  be the first such element of  $\mathfrak{J}$ 
8:     Find an essential  $\mathcal{T}$ -countermodel  $\mathcal{J}' \subseteq \mathcal{I} \times_r \mathcal{J}$  (entailment queries)
9:     replace  $\mathcal{J}$  in  $\mathfrak{J}$  with  $\mathcal{J}'$ 
10:  else
11:    append  $\mathcal{I}$  to  $\mathfrak{J}$ 
12:  end if
13:  Construct  $\mathcal{H} = \{C_{\mathcal{I}} \sqsubseteq A \mid \mathcal{I} \in \mathfrak{J}, \mathcal{T} \models C_{\mathcal{I}} \sqsubseteq A\}$  (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 [21] 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  $\mathcal{I} \times_r \mathcal{J}$  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 [21]: (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  $\mathcal{I}|_{\text{root}}^-$  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.  $\mathcal{I}|_{\text{root}}^- \models \mathcal{T}$ ;
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  $\mathcal{EL}_{\text{lhs}}$  TBoxes is given as Algorithm 2. In Line 6, we write  $\mathcal{J} \not\approx (\mathcal{I} \times_r \mathcal{J})$  as shorthand for  $(\mathcal{J}, \rho_{\mathcal{J}}) \not\approx (\mathcal{I} \times_r \mathcal{J}, (\rho_{\mathcal{I}}, \rho_{\mathcal{J}}))$ . 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  $\mathcal{I} \in \mathfrak{J}$  and  $A$  a concept name from  $\Sigma$ . We will later show that the length of  $\mathfrak{J}$  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  $\mathcal{J}' \subseteq \mathcal{I} \times_r \mathcal{J}$  using only polynomially many entailment queries in  $|\mathcal{T}| + |\mathcal{I}| + |\mathcal{J}|$ . Specifically,  $\mathcal{J}'$  is obtained from  $\mathcal{I} \times_r \mathcal{J}$  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  $\mathfrak{J}$  is bounded by the number of CIs in  $\mathcal{T}$  and
- (ii) each position of the sequence  $\mathfrak{J}$  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.**  *$\mathcal{EL}_{\text{IHS}}$  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  $\mathcal{EL}_{\text{IHS}}$  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. Zakharyashev. 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. McGuinness, 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  $\mathcal{EL}$ -GCI. 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. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyashev. The combined approach to query answering in DL-Lite. In *KR*, 2010.
20. J. Lehmann and P. Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203–250, 2010.
21. C. Lutz, R. Piro, and F. Wolter. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *IJCAI*, pages 983–988, 2011.
22. L. D. Raedt. Logical settings for concept-learning. *Artif. Intell.*, 95(1):187–201, 1997.
23. C. Reddy and P. Tadepalli. Learning Horn definitions: Theory and an application to planning. *New Generation Comput.*, 17(1):77–98, 1999.
24. B. ten Cate, V. Dalmau, and P. G. Kolaitis. Learning schema mappings. In *ICDT*, pages 182–195, 2012.

## A Proofs and examples for Section 3

We state Lemma 2 again.

**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 entailment queries in polynomial time in  $|C_1| + |C_2|$ .*

*Proof.* We apply the rule (sibling merging) from above exhaustively to  $A \sqsubseteq C_1 \sqcap C_2$  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) parent/child-merged for  $\mathcal{T}$ , and (iii) 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 readily show that an inclusion with properties (i)–(iii) still has those properties after applying the rule (sibling merging) above. Thus  $A \sqsubseteq C$  is  $\mathcal{T}$ -essential, as required.  $\square$

In our argument that the learning algorithm runs in polynomial time, we claimed that if  $A \sqsubseteq C$  is  $\mathcal{T}$ -essential,  $\mathcal{T} \models A \sqsubseteq C'$ , and  $\models C' \sqsubseteq C$ , then the tree corresponding to  $C$  is obtained from the tree corresponding to  $C'$  by removing subtrees. This follows from the following lemma.

**Lemma 8.** *Assume that  $A \sqsubseteq C$  is  $\mathcal{T}$ -essential,  $\mathcal{T} \models A \sqsubseteq C'$ , and  $\models C' \sqsubseteq C$ . Then there is an injective homomorphism  $h$  mapping the labelled tree  $T_C$  to the labeled tree  $T_{C'}$  such that*

- the root of  $T_C$  is mapped to the root of  $T_{C'}$ ;
- $A \in l(d)$  if  $A \in l(h(d))$ , for all concept names  $A$  and all  $d$  in  $T_C$ .

*Proof.* We can regard  $T_{C'}$  as an interpretation  $\mathcal{I}_{C'}$ . This interpretation satisfies  $C'$  in its root and so, since  $\models C' \sqsubseteq C$ , it satisfies  $C$  in its root. Thus, there is a homomorphism  $h$  from  $T_C$  to  $T_{C'}$  mapping the root of  $T_C$  to the root of  $T_{C'}$ .  $h$  is injective since  $A \sqsubseteq C$  is sibling and parent/child merged and  $\mathcal{T} \models A \sqsubseteq C'$ . Finally  $A \in l(d)$  if  $A \in h(l(d))$ , for all concept names  $A$  and all  $d$  in  $T_C$ , follows from concept saturatedness for  $\mathcal{T}$  of  $A \sqsubseteq C$  and  $\mathcal{T} \models A \sqsubseteq C'$ .  $\square$

We state Lemma 3 again.

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

*Proof.* The proof employs canonical models for DL-Lite TBoxes [19]. Assume  $A \sqsubseteq C$  is  $\mathcal{T}$ -essential and  $\mathcal{T} = \mathcal{H}_{basic} \cup \mathcal{T}'$ , where  $\mathcal{T}'$  consists of inclusions of the form  $E \sqsubseteq D$  in named form.

Let  $A \sqsubseteq D_1, \dots, A \sqsubseteq D_n$  be the list of all inclusions in  $\mathcal{T}$  with  $A$  on the left-hand side. We define the canonical model  $\mathcal{I}_{A, \mathcal{T}}$  as the union of a sequence of interpretations  $\mathcal{I}_0, \mathcal{I}_1, \dots$ . Let  $\mathcal{I}_0$  be the tree shaped interpretation defined by the labelled tree corresponding to  $D := D_1 \sqcap \dots \sqcap D_n$ . Its root is denoted by  $\rho_A$ . Now assume  $\mathcal{I}_n$  has been defined. To define  $\mathcal{I}_{n+1}$ ,

- let  $i \leq n$  be minimal such that there exists  $\exists r. \top \sqsubseteq E \in \mathcal{H}_{basic}$  with  $E$  a concept name and  $d \in (\exists r. \top)^{\mathcal{I}_i}$  with  $d \notin E^{\mathcal{I}_n}$ . Then define  $\mathcal{I}_{n+1}$  in the same way as  $\mathcal{I}_n$  except that  $E^{\mathcal{I}_{n+1}} := E^{\mathcal{I}_n} \cup \{d\}$ .
- Otherwise let  $i \leq n$  be minimal such that there exists  $E \sqsubseteq F \in \mathcal{T}$  with  $E$  a concept name and  $d \in E^{\mathcal{I}_i}$  such that  $d \notin F^{\mathcal{I}_n}$ . Then add to  $\mathcal{I}_n$  a copy of the tree shaped interpretation  $\mathcal{I}_{d,F}$  defined by the labelled tree corresponding to  $F$  and identify  $d$  with the root of  $\mathcal{I}_{d,F}$ .
- Otherwise let  $\mathcal{I}_{A,\mathcal{T}} := \mathcal{I}_n$ .

If Point 3 does not apply in the definition of the sequence  $\mathcal{I}_0, \mathcal{I}_1, \dots$ , then let  $\mathcal{I}_{A,\mathcal{T}} := \bigcup_{n \geq 0} \mathcal{I}_n$ .

In what follows we use the fact that for any interpretation  $\mathcal{J}$ ,  $d \in \Delta^{\mathcal{J}}$ , and  $\mathcal{ELI}$  concept  $F$ ,  $d \in F^{\mathcal{J}}$  iff there exists a homomorphism  $h$  from the labelled tree corresponding to  $F$  into  $\mathcal{J}$  with  $h(a_D) = d$ .  $\mathcal{I}_{A,\mathcal{T}}$  has the following properties:

1.  $\mathcal{I}_{A,\mathcal{T}}$  is a model of  $\mathcal{T}$ ;
2. for every  $\mathcal{ELI}$  concept  $F$ :  $\rho_A \in F^{\mathcal{I}_{A,\mathcal{T}}}$  iff  $\mathcal{T} \models A \sqsubseteq D$ ;
3. for any  $d \in \Delta^{\mathcal{I}_0}$  and  $\mathcal{ELI}$  concept  $F$ : if there exists a homomorphism  $h$  from the labelled tree corresponding to  $F$  into  $\mathcal{I}$  such that  $h(a_F) = d$  and all nodes in  $F$  distinct from the root  $a_F$  are mapped to  $\Delta^{\mathcal{I}} \setminus \Delta^{\mathcal{I}_0}$ , then there exists a concept name  $E$  with  $d \in E^{\mathcal{I}}$  such that  $\mathcal{T} \models E \sqsubseteq F$ .

We now show

Claim 1. There is an injective homomorphism from the labelled tree corresponding to  $C$  into the restriction  $\mathcal{J}$  of  $\mathcal{I}$  to  $\Delta^{\mathcal{I}_0}$  which maps  $a_C$  to  $\rho_A$ .

By Point 2 above and since  $\mathcal{T} \models A \sqsubseteq C$ , there is a homomorphism  $h$  from the labelled tree  $T_C$  corresponding to  $C$  into  $\mathcal{I}_{A,\mathcal{T}}$  which maps  $a_C$  to  $\rho_A$ . It follows from Point 2 again and the condition that  $A \sqsubseteq C$  is sibling and parent/child merged for  $\mathcal{T}$  that  $h$  is injective. It thus remains to be proved that  $h$  is into  $\mathcal{J}$  (rather than  $\mathcal{I}$ ). By concept saturatedness of  $A \sqsubseteq C$  and Point 2, we have (\*):  $h(d) \in E^{\mathcal{I}}$  iff  $E \in l(d)$  for every node  $d$  in  $T_C$  and every concept name  $E$ .

Assume now that  $h(d') \notin \Delta^{\mathcal{I}_0}$  for some node  $d'$  in  $T_C$ . Assume that  $d'$  has minimal distance from  $d_C$  with this property - thus the ancestor  $d$  of  $d'$  in  $T_C$  is mapped to  $\Delta^{\mathcal{I}_0}$ . Let  $l(d, d') = r$ . We make a case distinction:

- the subtree generated by  $d'$  is mapped into  $\Delta^{\mathcal{I}} \setminus \Delta^{\mathcal{I}_0}$ . Let  $C' = \exists r. C''$ , where  $C''$  corresponds to the subtree generated by  $d'$  in  $C$ . By Point 3, there exists a concept name  $E$  with  $h(d) \in E^{\mathcal{I}}$  such that  $\mathcal{T} \models E \sqsubseteq C'$ . By (\*),  $E \in l(d)$ . If  $d \neq \rho_A$  or  $A \neq E$ , then  $A \sqsubseteq C$  is not minimal for  $\mathcal{T}$  and we have derived a contradiction. If  $d = \rho_A$  and no  $E$  non- $\mathcal{T}$ -equivalent to  $A$  can be chosen, then  $d' \in \Delta^{\mathcal{I}_0}$  which is a contradiction.
- the subtree generated by  $d'$  is not mapped into  $\Delta^{\mathcal{I}} \setminus \Delta^{\mathcal{I}_0}$ . Then  $h$  is not injective and we have again a contradiction.

Claim 1 directly implies Lemma 3. □

We close this section by proving that if one (or more) of the four conditions for being  $\mathcal{T}$ -essential is removed, then Algorithm 1 requires at least exponential time or does not even terminate.

- Assume (concept saturatedness) is omitted from the definition of  $\mathcal{T}$ -essential. Let  $\mathcal{T} = \{A \sqsubseteq \exists r.A\}$ . Then the oracle can provide for then  $n$ th equivalence query in the while loop of Algorithm 1 the positive counterexamples  $A \sqsubseteq \exists r^{n+2}.\top$ . The algorithm does not terminate.
- Assume (sibling-merged) is omitted from the definition of  $\mathcal{T}$ -essential. Define concepts  $C_i$  as follows:

$$C_0 = \exists s_0.A_0 \sqcap \exists s_1.A_1, \quad C_{i+1} = C_0 \sqcap \exists r.C_i$$

and let  $\mathcal{T}_n = \{A \sqsubseteq \exists s.C_n\}$ . For  $M \subseteq \{0, \dots, n\}$  set  $C_0^M = \exists s_0.A_0$  if  $0 \in M$  and  $C_0^M = \exists s_1.A_1$  if  $0 \notin M$ . Also, let  $C_{i+1}^M = \exists s_0.A_0 \sqcap \exists r.C_i^M$  if  $i+1 \in M$  and  $C_{i+1}^M = \exists s_1.A_1 \sqcap \exists r.C_i^M$  if  $i+1 \notin M$ . Then the oracle can provide for the first  $2^{n+1}$  equivalence queries in the while loop of Algorithm 1 a fresh positive counterexamples  $A \sqsubseteq \exists s.C_n^M$  by always choosing a fresh  $M \subseteq \{0, \dots, n\}$ .

- Assume (parent/child-merged) is omitted from the definition of  $\mathcal{T}$ -essential. Let

$$\mathcal{T}_n = \{A \sqsubseteq \exists r_1.\top \sqcap \dots \sqcap \exists r_n.\top \sqcap \exists r_1^-\top \sqcap \dots \sqcap \exists r_n^-\top \sqcap \exists s.A_1 \sqcap \dots \sqcap \exists s.A_n \sqcap \exists s.B_1 \sqcap \dots \sqcap \exists s.B_n\}$$

Define for any  $M \subseteq \{1, \dots, n\}$  (with  $\overline{M} = \{1, \dots, n\} \setminus M$ ),

$$C_M = \exists s_0.\exists t_1.\exists t_1^-\dots.\exists t_n.\exists t_n^-. \prod_{i \in M} \exists s.A_i \sqcap \prod_{i \in \overline{M}} \exists s.B_i,$$

where  $t_i = r_i$  if  $i \in M$  and  $t_i = r_i^-$  if  $i \in \overline{M}$ . Then the oracle can provide for the first  $2^{n+1}$  equivalence queries in the while loop of Algorithm 1 a fresh positive counterexample  $A \sqsubseteq C_M$ , for  $M \subseteq \{0, \dots, n\}$ .

- Assume (minimization) is omitted from the definition of  $\mathcal{T}$ -essential. Let  $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq \exists r.B\}$ . Then the oracle can provide for the  $n$ th equivalence query in the while loop of Algorithm 1 the positive counterexamples  $A \sqsubseteq \exists r^{n+2}.\top$ . The algorithm does not terminate.

## B Proofs for Section 4

To prove Lemma 4, we first show two technical lemmas. We also require the following lemma from [18] that characterizes concept inclusions entailed by acyclic  $\mathcal{EL}$  TBoxes.

**Lemma 9.** *Let  $\mathcal{T}$  be an acyclic  $\mathcal{EL}$ -TBox,  $r$  a role name and  $D$  an  $\mathcal{EL}$ -concept. Suppose that  $\mathcal{T} \models \prod_{1 \leq i \leq n} A_i \sqcap \prod_{1 \leq j \leq m} \exists r_j.C_j \sqsubseteq D$ , where  $A_i$  are concept names for  $1 \leq i \leq n$ ,  $C_j$  are  $\mathcal{EL}$ -concepts for  $1 \leq j \leq m$ , and  $m, n \geq 0$ , then*

- *if  $D$  is a concept name such that  $\mathcal{T}$  does not contain an inclusion  $D \equiv C$ , for some concept  $C$ , then there exists  $A_i$ ,  $1 \leq i \leq n$ , such that  $\mathcal{T} \models A_i \sqsubseteq D$ ;*

- if  $D$  is of the form  $\exists r.D'$  then either (i) there exists  $A_i$ ,  $1 \leq i \leq n$ , such that  $\mathcal{T} \models A_i \sqsubseteq \exists r.D'$  or (ii) there exists  $r_j$ ,  $1 \leq j \leq m$ , such that  $r_j = r$  and  $\mathcal{T} \models C_j \sqsubseteq D'$ .

**Lemma 10.** For any  $0 \leq m \leq n$ , any sequence of role names  $\sigma = \sigma^1 \dots \sigma^m$ , any  $L = (\sigma_1, \dots, \sigma_n) \in \mathfrak{L}_n$  and any  $\mathcal{EL}$ -concept  $C$  over  $\Sigma_n$ : if  $\mathcal{T}_L \models C \sqsubseteq \exists \sigma.M$  then either

1.  $m = n$ ,  $\sigma = \sigma_i$ , for some  $1 \leq i \leq n$  and  $C$  is of the form  $A \sqcap C'$ ,  $A_i \sqcap C'$  or  $B_i \sqcap C'$ , for some  $\mathcal{EL}$ -concept  $C'$ ; or
2.  $\models C \sqsubseteq \exists \sigma.M$ .

*Proof.* We prove the proposition by induction on  $m$ . If  $m = 0$ , by Lemma 9,  $C$  is of the form  $Z \sqcap C'$ , for some concept name  $Z$  and concept  $C'$ , and  $\mathcal{T}_L \models Z \sqsubseteq M$ . This is only possible if  $Z$  is  $M$  itself, so  $\models C \sqsubseteq M$ .

Let  $m > 0$ . By Lemma 9 we have one of the following two cases:

- $C$  is of the form  $X \sqcap C'$ , for some concept name  $X$  and concept  $C'$  such that  $\mathcal{T}_L \models X \sqsubseteq \exists \sigma.M$ . It is easy to see that this is only possible if  $m = n$ ,  $\sigma = \sigma_i$  and  $X$  is one of  $A$ ,  $A_i$  or  $B_i$ .
- $C$  is of the form  $\exists \sigma^1.C' \sqcap C''$  for some concepts  $C$ ,  $C'$  such that  $\mathcal{T}_L \models C' \sqsubseteq \exists \sigma^2 \dots \exists \sigma^m.M$ . By induction hypothesis,  $\models C' \sqsubseteq \exists \sigma^2 \dots \exists \sigma^m.M$  but then  $\models C \sqsubseteq \exists \sigma.M$ .

□

**Lemma 11.** For any acyclic  $\mathcal{EL}$  TBox  $\mathcal{T}$ , any inclusion  $A \sqsubseteq C \in \mathcal{T}$  and any concept of the form  $\exists t.D$  we have  $\mathcal{T} \models A \sqsubseteq \exists t.D$  if, and only if,  $\mathcal{T} \models C \sqsubseteq \exists t.D$ .

*Proof (sketch).* As  $A \sqsubseteq C \in \mathcal{T}$ , if  $\mathcal{T} \models C \sqsubseteq \exists t.D$  we have  $\mathcal{T} \models A \sqsubseteq \exists t.D$ . The converse direction is less trivial but can be proved by manipulating models using the condition that  $\mathcal{T}$  is acyclic. □

We are now ready to prove Lemma 4.

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

- 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  $|C|$ .

*Proof.* We prove the lemma by induction on the size of  $D$ . We assume throughout the proof that there exists some  $L_0 \in \mathfrak{L}_n$  such that  $\mathcal{T}_{L_0} \models C \sqsubseteq D$ .

*Base case:*  $D$  is a concept name. Consider possibilities.

- $D$  is one of  $X_i$ ,  $A_i$ ,  $B_i$  or  $M$ , for  $i \geq 1$ . By Lemma 9,  $C$  is of the form  $Z \sqcap C'$ , for some concept name  $Z$ , and  $\mathcal{T}_{L_0} \models Z \sqsubseteq D$ . For  $D$  being one of  $X_i$ ,  $A_i$ ,  $B_i$  or  $M$  such  $Z$  can only be  $D$  itself. But then for every  $L \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models C \sqsubseteq D$ .
- $D$  is  $X_0$ . By Lemma 9,  $C$  is of the form  $Z \sqcap C'$ , for some concept name  $Z$ , and  $\mathcal{T}_{L_0} \models Z \sqsubseteq X_0$ . This is the case if either  $Z$  is  $X_0$  itself, or  $Z$  is one of  $A$ ,  $A_1, B_1, \dots, A_n, B_n$ . In either case, for every  $L \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models C \sqsubseteq X_0$ .

- $D$  is  $A$ . If  $C$  is form  $A \sqcap C'$  or, for every  $i : 1 \leq i \leq n$ ,  $A_i$  or  $B_i$  is a conjunct of  $C$ , for every  $L \in \mathfrak{L}_n$  we clearly have  $\mathcal{T}_L \models C \sqsubseteq A$ . Suppose that this is not so. Then for some  $j : 1 \leq j \leq n$ ,  $C$  is neither of the form  $A \sqcap C'$  nor  $A_j \sqcap C'$  nor  $B_j \sqcap C'$ . Let  $L = (\sigma_1, \dots, \sigma_n) \in \mathfrak{L}_n$  be such that  $\mathcal{T}_L \models C \sqsubseteq A$ . Notice that  $\mathcal{T}_L \models C \sqsubseteq A$ , for  $L = (\sigma_1, \dots, \sigma_n) \in \mathfrak{L}_n$ , if, and only if,  $\mathcal{T}_L \models C \sqsubseteq X_0 \sqcap \exists \sigma_1.M \sqcap \dots \sqcap \exists \sigma_n.M$ . By Lemma 10, for such a  $\mathcal{T}_L$  we must have  $\models C \sqsubseteq \exists \sigma_j.M$ . Clearly, the number of different  $L = (\sigma_1, \dots, \sigma_n) \in \mathfrak{L}_n$  with  $\models C \sqsubseteq \exists \sigma_j.M$  does not exceed  $|C|$ . Thus, either for every  $L \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models C \sqsubseteq A$  or the number of different  $L \in \mathfrak{L}_n$  such that  $\mathcal{T}_L \models C \sqsubseteq A$  does not exceed  $|C|$ .

*Induction step.* If  $D = D_1 \sqcap D_2$ , then  $\mathcal{T}_L \models C \sqsubseteq D$  if, and only if,  $\mathcal{T} \models C \sqsubseteq D_i$ ,  $i = 1, 2$ . By induction hypothesis, for  $i = 1, 2$  either for every  $L \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models C \sqsubseteq D_i$ , or there exist at most  $|C|$  different  $L \in \mathfrak{L}_n$  such that  $\mathcal{T}_L \models C \sqsubseteq D_i$ . Thus 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  $\mathcal{T}_L \models C \sqsubseteq D$  also does not exceed  $|C|$ .

Let  $D = \exists t.D'$ . Suppose that for some  $L \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models C \sqsubseteq D$ . Then, by Lemma 9, either there exists a conjunct  $Z$  of  $C$ ,  $Z$  a concept name, such that  $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$  or there exists a conjunct  $\exists t.C'$  of  $C$  with  $\mathcal{T}_L \models C' \sqsubseteq D'$ . We analyse for every conjunct of  $C$  of the form  $Z$  or  $\exists t.C'$  for how many different  $L \in \mathfrak{L}_n$  it holds that  $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$  (or  $\mathcal{T}_L \models \exists t.C' \sqsubseteq \exists t.D'$  respectively).

- Let  $Z$  be a conjunct of  $C$  such that  $Z$  is a concept name and  $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$ . Notice that  $Z$  cannot be  $M$  as for no  $L \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models M \sqsubseteq \exists t.D'$ . Consider the remaining possibilities.
  - $Z$  is one of  $X_i$ ,  $i \geq 0$ . It is easy to see that for  $L, L' \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models X_i \sqsubseteq \exists t.D'$  if, and only if  $\mathcal{T}_{L'} \models X_i \sqsubseteq \exists t.D'$ . Thus, for every  $L \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$ .
  - $Z$  is one of  $A_i, B_i$  for  $i \geq 1$ . By Lemma 11,  $\mathcal{T}_L \models Z \sqsubseteq \exists t.D'$  if, and only if,  $\mathcal{T}_L \models X_0 \sqcap \exists \sigma_i.M \sqsubseteq \exists t.D'$ . By Lemma 9, either  $\mathcal{T}_L \models X_0 \sqsubseteq \exists t.D'$  or  $\mathcal{T}_L \models \exists \sigma_i.M \sqsubseteq \exists t.D'$ . If  $\mathcal{T}_L \models X_0 \sqsubseteq \exists t.D'$  then for every  $L \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models C \sqsubseteq \exists t.D'$ . Suppose that  $\exists t.D'$  is such that  $\mathcal{T}_L \not\models X_0 \sqsubseteq \exists t.D'$  and  $\mathcal{T}_L \models \exists \sigma_i.M \sqsubseteq \exists t.D'$ . This is only possible when  $\exists t.D'$  is  $\exists \sigma_i.M$ . Notice that all  $\sigma_i$  are unique so there exists exactly one  $L \in \mathfrak{L}_n$  (namely,  $L$  is  $L_0$ ) such that  $\mathcal{T}_L \models Z \sqsubseteq \exists \sigma_i.M$ .
  - $Z$  is  $A$ . Suppose that for some  $L = (\sigma_1, \dots, \sigma_n) \in \mathfrak{L}_n$  we have  $\mathcal{T}_L \models A \exists t.D'$ , equivalently  $\mathcal{T}_L \models X_0 \sqcap \exists \sigma_1.M \sqcap \dots \sqcap \sigma_n.M \sqsubseteq \exists t.D'$ . By Lemma 9, either  $\mathcal{T}_L \models X_0 \sqsubseteq \exists t.D'$  or  $\mathcal{T}_L \models \exists \sigma_i.M \sqsubseteq \exists t.D'$ , for some  $i : 1 \leq i \leq n$ , so, as above, unless  $\mathcal{T}_L \models X_0 \sqsubseteq \exists t.D'$  we have  $\exists t.D'$  is  $\exists \sigma_i.M$ . But then  $L = L_0$ .
- Let  $\exists t.C'$  be a conjunct of  $C$  with  $\mathcal{T}_L \models C' \sqsubseteq D'$ . The induction hypothesis implies that the number of different  $L \in \mathfrak{L}_n$  such that  $\mathcal{T}_L \models C' \sqsubseteq D'$  does not exceed  $|C'|$ .

To summarize, either  $\mathcal{T}_L \models C \sqsubseteq \exists t.D'$  for every  $L \in \mathfrak{L}_n$  or for every conjunct  $C_0$  of  $C$  of the form  $Z$  or  $\exists t.C'$ , the number of different  $L \in \mathfrak{L}_n$  such that  $\mathcal{T}_L \models C_0 \sqsubseteq \exists t.D'$  does not exceed  $|C_0|$ . Hence the number of different  $L \in \mathfrak{L}_n$  such that  $\mathcal{T}_L \models C \sqsubseteq \exists t.D'$  does not exceed  $|C|$ .  $\square$

The next lemma prepares the proof of Lemma 5.

**Lemma 12.** *For any  $i \geq 0$  and  $\Sigma_n$ -concept  $D$ , if  $\mathcal{T}_0 \not\models X_i \sqsubseteq D$  then there exists a sequence of role names  $t_1, \dots, t_l$  such that  $\models D \sqsubseteq \exists t_1 \dots \exists t_l.Y$  and  $\mathcal{T}_0 \not\models X_i \sqsubseteq \exists t_1 \dots \exists t_l.Y$ , where  $Y$  is either  $\top$  or a concept name,  $0 \leq l \leq n - i + 1$ .*

*Proof.* We prove the lemma by induction on  $i$ . The base case is  $i = n$ . Then  $\mathcal{T}_0 \not\models X_n \sqsubseteq D$  either if  $\models D \sqsubseteq \exists t.\top$ , for some role name  $t$ , or  $\models D \sqsubseteq Y$ , for some concept name  $Y \neq X_n$ .

Suppose that the proposition is proved for  $0 < j \leq n$  and let  $i = j - 1$ . We proceed by induction on the structure of  $D$ . If  $D$  is a concept name, we are done as for no concept name  $Z \neq X_i$  we have  $\mathcal{T}_0 \models X_i \sqsubseteq Z$ . If  $D$  is of the form  $\exists t.D'$ , where  $t$  is one of  $r, s$ , we, obviously, have  $\mathcal{T}_0 \not\models X_{i+1} \sqsubseteq D'$ , and so, by induction hypothesis, there exists a sequence of role names  $t_1, \dots, t_l$ , with  $l \leq n - i$ , such that  $\mathcal{T}_0 \not\models X_{i+1} \sqsubseteq \exists t_1 \dots \exists t_l.Y$  and  $\models D' \sqsubseteq \exists t_1 \dots \exists t_l.Y$ . But then, by Lemma 11 and Lemma 9,  $\mathcal{T}_0 \not\models X_i \sqsubseteq \exists t.t_1 \dots \exists t_l.Y$  and  $\models \exists t.D' \sqsubseteq \exists t.t_1 \dots \exists t_l.Y$ . If  $D$  is of the form  $D = D_1 \sqcap D_2$ , for one of  $D_i, i = 1, 2$ , we have  $\mathcal{T}_0 \not\models X_i \sqsubseteq D_i$  and the proposition holds by induction.  $\square$

**Lemma 5** *For any  $n > 0$  and any  $\mathcal{EL}$  TBox  $\mathcal{H}$  in  $\Sigma_n$  with  $|\mathcal{H}| < 2^n$  there exists a  $\Sigma_n$ -concept inclusion  $C \sqsubseteq D$  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  $\mathcal{T}_L \not\models C \sqsubseteq D$ .*

*Proof.* We define a TBox  $\mathcal{T}_\mathbf{b}$  and prove that one can select the required  $\mathcal{EL}$  inclusion  $C \sqsubseteq D$  in such a way that either  $\mathcal{H} \models C \sqsubseteq D$  and  $\mathcal{T}_\mathbf{b} \not\models C \sqsubseteq D$ , or vice versa.

To define  $\mathcal{T}_\mathbf{b}$ , for any sequence  $\mathbf{b} = b_1 \dots b_n$ , where every  $b_i$  is either 0 or 1, we denote by  $C_\mathbf{b}$  the conjunction  $\bigwedge_{i \leq n} C_i$ , where  $C_i = A_i$  if  $b_i = 1$  and  $C_i = B_i$  if  $b_i = 0$ . Then we define

$$\mathcal{T}_\mathbf{b} = \mathcal{T}_0 \cup \{C_\mathbf{b} \sqsubseteq A \sqcap X_0 \mid \mathbf{b} \in \{0, 1\}^n\}.$$

Consider possibilities for  $\mathcal{H}$  and  $\mathcal{T}_\mathbf{b}$ .

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

(2) Suppose that for some  $\mathbf{b} \in \{0, 1\}^n$  and a concept of the form  $\exists t.D'$  we have  $\mathcal{H} \models C_\mathbf{b} \sqsubseteq \exists t.D'$  and  $\mathcal{T}_\mathbf{b} \not\models C_\mathbf{b} \sqsubseteq \exists t.D'$ . To ‘minimise’  $C_\mathbf{b} \sqsubseteq \exists t.D'$ , notice that  $\mathcal{T}_0 \not\models X_0 \sqsubseteq \exists t.D'$ . Then, by Lemma 12, there exists a sequence of role names  $t_1, \dots, t_l$ , for  $0 \leq l \leq n + 1$  and  $Y$  being  $\top$  or a concept name such that  $\models \exists t.D' \sqsubseteq \exists t_1 \dots \exists t_l.Y$ , so  $\mathcal{H} \models C_\mathbf{b} \sqsubseteq \exists t_1 \dots \exists t_l.Y$ , and  $\mathcal{T}_0 \not\models X_0 \sqsubseteq \exists t_1 \dots \exists t_l.Y$ . Clearly, the size of  $C_\mathbf{b} \sqsubseteq \exists t_1 \dots \exists t_l.Y$  does not exceed  $6n$ . It remains to prove that  $\mathcal{T}_L \models C_\mathbf{b} \sqsubseteq \exists t_1 \dots \exists t_l.Y$  for at most one  $L \in \mathcal{L}_n$ .

Suppose for some  $L \in \mathcal{L}_n$  we have  $\mathcal{T}_L \models C_\mathbf{b} \sqsubseteq \exists t_1 \dots \exists t_l.Y$ . By Lemma 9, there exists  $A_j$  or  $B_j$  such that  $\mathcal{T}_L \models A_j \sqsubseteq \exists t_1 \dots \exists t_l.Y$  (or  $\mathcal{T}_L \models B_j \sqsubseteq \exists t_1 \dots \exists t_l.Y$ , respectively). As  $\mathcal{T}_0 \not\models X_0 \sqsubseteq \exists t_1 \dots \exists t_l.Y$  it is easy to see that this is only possible when  $l = n$ ,  $(t_1, t_2, \dots, t_n) = \sigma_j$ , and  $Y$  is  $M$ . Since every  $\sigma_j$  is unique, for every  $L' \in \mathcal{L}_n$  such that  $L' \neq L$  we have  $\mathcal{T}_{L'} \not\models C_\mathbf{b} \sqsubseteq \exists \sigma_j.M$ .

Thus,  $C_b \sqsubseteq \exists t_1 \dots \exists t_l.Y$  is as required.

(3) Finally, suppose that Case 1 and 2 above do not apply. Then  $\mathcal{H} \models \mathcal{T}_\cap$  and for every  $\mathbf{b} \in \{0, 1\}^n$  and every  $\mathcal{EL}$ -concept over  $\Sigma_n$  of the form  $\exists t.D'$ : if  $\mathcal{H} \models C_b \sqsubseteq \exists t.D'$  then  $\mathcal{T}_0 \models X_0 \sqsubseteq \exists t.D'$ . We show that unless there exists an inclusion  $C \sqsubseteq D$  satisfying the conditions of the lemma,  $\mathcal{H}$  contains at least  $2^n$  different inclusions. Thus, we have derived a contradiction.

Fix  $\mathbf{b} \in \{0, 1\}^n$ . As  $\mathcal{H} \models \mathcal{T}_\cap$  we have  $\mathcal{H} \models C_b \sqsubseteq A$ . Then there must exist an (at least one) inclusion  $C \sqsubseteq A \sqcap D \in \mathcal{H}$  such that  $\mathcal{H} \models C_b \sqsubseteq C$  and  $\not\models C \sqsubseteq A$ . Let  $C = Z_1 \sqcap \dots \sqcap Z_m \sqcap \exists t_1.C'_1 \sqcap \dots \sqcap \exists t_l.C'_l$ , where  $Z_1, \dots, Z_m$  are different concept names. As  $\mathcal{H} \models C_b \sqsubseteq \exists t_j.C'_j$  we have  $\mathcal{T}_0 \models X_0 \sqsubseteq \exists t_j.C'_j$ , for  $j = 1, \dots, l$ . As  $\mathcal{H} \models \mathcal{T}_\cap$  we have  $\mathcal{H} \models X_0 \sqsubseteq \exists t_j.C'_j$ , for  $j = 1, \dots, l$ . So  $\mathcal{H} \models Z_1 \sqcap \dots \sqcap Z_m \sqcap X_0 \sqsubseteq A$ .

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

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

Assume that  $Z_0 \sqcap \dots \sqcap Z_m \sqcap X_0$  contains a conjunct  $A_i$  such that  $b_i \neq 1$ . Then  $\mathcal{H} \models C_b \sqsubseteq A_i$  and for no  $L \in \mathcal{L}_n$  we have  $\mathcal{T}_L \models C_b \sqsubseteq A_i$ . The size of  $C_b \sqsubseteq A_i$  does not exceed  $6n$ , so it is as required.

The only remaining option is that  $Z_1 \sqcap \dots \sqcap Z_m \sqcap X_0$  contains exactly the  $A_i$  with  $b_i = 1$  and exactly the  $B_i$  with  $b_i = 0$ .

This argument applies to arbitrary  $\mathbf{b} \in \{0, 1\}^n$ . Thus if there exists no inclusion  $C \sqsubseteq D$  satisfying the conditions of the lemma then  $\mathcal{H}$  contains at least  $2^n$  inclusions.  $\square$

## C Proofs for Section 5

**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  $\mathcal{I} \models \mathcal{H}$  only polynomially many entailment queries in  $|\mathcal{T}| + |C|$ .

*Proof.* Let  $C \sqsubseteq A$  be a positive counterexample for  $\mathcal{T}$  relative to  $\mathcal{H}$ . Since  $\mathcal{H} \not\models C \sqsubseteq A$ , we know that  $A$  does not occur as a top-level conjunct in  $C$ . Consequently,  $\rho_{\mathcal{I}_C} \in C^{\mathcal{I}_C} \setminus A^{\mathcal{I}_C}$  and thus  $\mathcal{I}_C \not\models \mathcal{T}$ .

We can safely assume that  $\mathcal{I}_C \models \mathcal{H}$ . In fact, if this is not the case, let  $\mathcal{J}$  be obtained from  $\mathcal{I}_C$  by applying the CIs in  $\mathcal{H}$  as saturation rules. Then we can use  $C_{\mathcal{J}} \sqsubseteq A$  as a positive counterexample for  $\mathcal{T}$  relative to  $\mathcal{H}$  instead of  $C \sqsubseteq A$ . In particular,  $\emptyset \models C_{\mathcal{J}} \sqsubseteq C$ , and thus  $\mathcal{T} \models C \sqsubseteq A$  implies  $\mathcal{T} \models C_{\mathcal{J}} \sqsubseteq A$ . Moreover, it can be shown that  $\rho_{\mathcal{J}} \in B^{\mathcal{J}}$  implies  $\mathcal{H} \models C \sqsubseteq B$  for all concept names  $B$ . Consequently and since  $\mathcal{H} \not\models C \sqsubseteq A$ , we have  $\rho_{\mathcal{J}} \notin A^{\mathcal{J}}$ . Since  $\mathcal{J}$  is a model of  $\mathcal{H}$  and  $\rho_{\mathcal{J}} \in (C_{\mathcal{J}})^{\mathcal{J}}$ , we obtain  $\mathcal{H} \not\models C_{\mathcal{J}} \sqsubseteq A$ .

For an interpretation  $\mathcal{I}$  and an element  $d \in \Delta^{\mathcal{I}}$ , we use  $\mathcal{I}|_d$  to denote the subtree-interpretation of  $\mathcal{I}$  rooted at  $d$ . Choose an element  $d$  of  $\mathcal{I}_C$  such that  $\mathcal{I}_C|_d \not\models \mathcal{T}$  and the depth of  $d$  in the tree  $\mathcal{I}_C$  is maximal with this property. Such a  $d$  exists since  $\mathcal{I}_C \not\models \mathcal{T}$ . Let  $\mathcal{I} = \mathcal{I}_C|_d$ . Note that  $\mathcal{I} \not\models \mathcal{T}$  and, due to the choice of  $d$ , we have  $\mathcal{I}|_{\text{root}} \models \mathcal{T}$ .

As the next step, exhaustively apply the following operation: if  $d \in \Delta^{\mathcal{I}} \setminus \{\rho_{\mathcal{I}}\}$  and  $\mathcal{I}|_{d\downarrow} \not\models \mathcal{T}$  (which can be checked using an entailment query), then delete the subtree in  $\mathcal{I}$  rooted at  $d$ , that is, replace  $\mathcal{I}$  with  $\mathcal{I}|_{d\downarrow}$ . By construction, the interpretation  $\mathcal{I}$  ultimately obtained is a  $\mathcal{T}$ -countermodel. Moreover, it is essential:

1.  $\mathcal{I}|_{\text{root}} \models \mathcal{T}$  holds since it was true for the interpretation  $\mathcal{I}$  chosen initially and being a model of  $\mathcal{T}$  is preserved under removing subtrees.
2.  $\mathcal{I}|_{d\downarrow} \models \mathcal{T}$  for all  $d \in \Delta^{\mathcal{I}} \setminus \{\rho_{\mathcal{I}}\}$  since, otherwise, the subtree rooted at  $d$  would have been removed during the elimination step in the construction of  $\mathcal{I}$ .

It remains to note that  $\mathcal{I} \models \mathcal{H}$ . In fact,  $\mathcal{I}_C \models \mathcal{H}$  clearly implies  $\mathcal{I}_C|_d \models \mathcal{H}$  and removing subtrees in the further construction preserves being a model of  $\mathcal{H}$ .  $\square$

**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  $\mathcal{J}' \subseteq \mathcal{I} \times_r \mathcal{J}$  using only polynomially many entailment queries in  $|\mathcal{T}| + |\mathcal{I}| + |\mathcal{J}|$ . Specifically,  $\mathcal{J}'$  is obtained from  $\mathcal{I} \times_r \mathcal{J}$  by removing subtrees.

*Proof.* Let  $\mathcal{I}$  and  $\mathcal{J}$  be essential  $\mathcal{T}$ -countermodels with  $\mathcal{I} \times_r \mathcal{J} \not\models \mathcal{T}$ . Modify  $\mathcal{J}' := \mathcal{I} \times_r \mathcal{J}$  by exhaustively applying the following operation: if  $d \in \Delta^{\mathcal{J}'} \setminus \{\rho_{\mathcal{J}'}\}$  and  $\mathcal{J}'|_d \not\models \mathcal{T}$  (which is verified using an entailment query), then remove the subtree rooted at  $d$ , that is, replace  $\mathcal{J}'$  with  $\mathcal{J}'|_{d\downarrow}$ . By construction, the new  $\mathcal{J}'$  is a  $\mathcal{T}$ -countermodel. Moreover, it is essential:

- $\mathcal{J}'|_{\text{root}} \models \mathcal{T}$ .  
 $\mathcal{I}|_{\text{root}} \models \mathcal{T}$  and  $\mathcal{J}|_{\text{root}} \models \mathcal{T}$  implies  $\mathcal{I}|_{\text{root}} \times \mathcal{J}|_{\text{root}} \models \mathcal{T}$  and thus it remains to observe that  $\mathcal{J}'|_{\text{root}}$  can be obtained from  $\mathcal{I}|_{\text{root}} \times \mathcal{J}|_{\text{root}}$  by removing subtrees.
- $\mathcal{J}'|_{d\downarrow} \models \mathcal{T}$  for all  $d \in \Delta^{\mathcal{J}'} \setminus \{\rho_{\mathcal{J}'}\}$ .  
Otherwise, the subtree rooted at  $d$  would have been removed during the elimination step in the construction of  $\mathcal{J}'$ .  $\square$

We show the following two statements:

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

For easy reference, 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) above, we closely follow [4] 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.

In fact, Point (iii) is immediate since whenever a new  $\mathcal{I}_i$  is added to  $\mathfrak{J}$  in the algorithm, then  $\mathcal{I}_i$  is a  $\mathcal{T}$ -countermodel. To prove Point (iv), we first establish the intermediate Lemma 13 below. For a tree-shaped interpretation  $\mathcal{I}$  and a concept inclusion  $C \sqsubseteq A$ , we write  $\mathcal{I} \models^r C \sqsubseteq A$  if  $\rho_{\mathcal{I}} \notin C^{\mathcal{I}}$  or  $\rho_{\mathcal{I}} \in A^{\mathcal{I}}$ ; that is, the inclusion  $C \sqsubseteq A$  is satisfied at the root of  $\mathcal{I}$ , but not necessarily at other points in  $\mathcal{I}$ . It is easy to see that if some interpretation  $\mathcal{I}$  is a  $\mathcal{T}$ -countermodel, then there is a  $C \sqsubseteq A \in \mathcal{T}$  such that  $\mathcal{I} \not\models^r C \sqsubseteq A$ .

**Lemma 13.** *If the interpretation  $\mathcal{I}$  constructed in Line 5 of the algorithm satisfies  $\mathcal{I} \not\models^r C \sqsubseteq A \in \mathcal{T}$  and  $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$  for some  $j$ , then  $\mathcal{J} = \mathcal{I}_i$  is replaced with  $\mathcal{J}'$  in Line 9 for some  $i \leq j$ .*

*Proof.* Assume that the interpretation  $\mathcal{I}$  constructed in Line 5 of the algorithm satisfies  $\mathcal{I} \not\models^r C \sqsubseteq A \in \mathcal{T}$  and that there is some  $j$  with  $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$ . If there is some  $i < j$  such that  $\mathcal{I}_i \not\Rightarrow (\mathcal{I} \times_r \mathcal{I}_i)$  and  $\mathcal{I} \times_r \mathcal{I}_i \not\models \mathcal{T}$ , then  $\mathcal{J} = \mathcal{I}_{i'}$  will be replaced with  $\mathcal{J}'$  in Line 9 for some  $i' \leq i$  and we are done. Thus assume that there is no such  $i$ . We aim to show that  $\mathcal{J} = \mathcal{I}_j$  is replaced with  $\mathcal{J}'$  in Line 9. To this end, it suffices to prove that  $\mathcal{I}_j \not\Rightarrow (\mathcal{I} \times_r \mathcal{I}_j)$  and  $\mathcal{I} \times_r \mathcal{I}_j \not\models \mathcal{T}$ . The latter is a consequence of  $\mathcal{I} \not\models^r C \sqsubseteq A$  and  $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$ .

Assume to the contrary of what we have to show that  $\mathcal{I}_j \Rightarrow (\mathcal{I} \times_r \mathcal{I}_j)$ . We establish a contradiction against  $\mathcal{I} \models \mathcal{H}$  (which holds by construction of  $\mathcal{I}$  in the algorithm) by showing that

1.  $\mathcal{I} \not\models^r C_{\mathcal{I}_j} \sqsubseteq A$  and
2.  $C_{\mathcal{I}_j} \sqsubseteq A \in \mathcal{H}$ .

For Point 1,  $\mathcal{I}_j \Rightarrow (\mathcal{I} \times_r \mathcal{I}_j)$  and  $\rho_{\mathcal{I}_j} \in (C_{\mathcal{I}_j})^{\mathcal{I}_j}$  imply  $\rho_{\mathcal{I} \times_r \mathcal{I}_j} \in (C_{\mathcal{I}_j})^{\mathcal{I} \times_r \mathcal{I}_j}$ , which gives  $\rho_{\mathcal{I}} \in (C_{\mathcal{I}_j})^{\mathcal{I}}$ . It remains to observe that  $\mathcal{I} \not\models^r C \sqsubseteq A$  implies  $\rho_{\mathcal{I}} \notin A^{\mathcal{I}}$ .

In view of the construction of  $\mathcal{H}$  in the algorithm, Point 2 can be established by showing that  $\mathcal{T} \models C_{\mathcal{I}_j} \sqsubseteq A$ . Since  $C \sqsubseteq A \in \mathcal{T}$ , it suffices to prove that  $C_{\mathcal{I}_j} \sqsubseteq C$ . This, however, is an immediate consequence of the fact that  $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$  and the definition of  $C_{\mathcal{I}_j}$ .  $\square$

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

**Lemma 14.** *At any time of the algorithm execution, the following condition holds: if  $\mathcal{I}_i \not\models^r C \sqsubseteq A \in \mathcal{T}$  and  $j < i$ , then  $\rho_{\mathcal{I}_j} \notin C^{\mathcal{I}_j}$ .*

*Proof.* We prove the invariant formulated in Lemma 14 by induction on the number of iterations of the while loop. Clearly, the invariant is satisfied before the loop is entered. We now consider the two places where  $\mathfrak{J}$  is modified, that is, Line 9 and Line 11, starting with the latter.

In Line 11,  $\mathcal{I}$  is appended to  $\mathfrak{J}$ . Assume that  $\mathcal{I} \not\models^r C \sqsubseteq A \in \mathcal{T}$ . We have to show that, before  $\mathcal{I}$  was added to  $\mathfrak{J}$ , there was no  $\mathcal{I}_i \in \mathfrak{J}$  with  $\rho_{\mathcal{I}_i} \in C^{\mathcal{I}_j}$ . This, however, is immediate by Lemma 13.

Now assume that  $\mathcal{J}$  was replaced in Line 9 with  $\mathcal{J}'$ . We have to show two properties:

1. If  $\mathcal{J}' = \mathcal{I}_i \not\models^r C \sqsubseteq A \in \mathcal{T}$  and  $j < i$ , then  $\rho_{\mathcal{I}_j} \notin C^{\mathcal{I}_j}$ .  
Assume to the contrary that  $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$ . Since  $\mathcal{J}'$  is obtained from  $\mathcal{I} \times \mathcal{J}$  by removing subtrees (see Lemma 7),  $\mathcal{J}' \not\models^r C \sqsubseteq A$  implies  $\mathcal{I} \times \mathcal{J} \not\models^r C \sqsubseteq A$ . Consequently,  $\mathcal{I} \not\models^r C \sqsubseteq A$  or  $\mathcal{J} \not\models^r C \sqsubseteq A$ . The former and  $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$  yields  $i \leq j$  by Lemma 13, in contradiction to  $j < i$ . In the latter case, since  $\mathcal{I}_i = \mathcal{J}$  before the replacement of  $\mathcal{J}$  with  $\mathcal{J}'$ , we have a contradiction against the induction hypothesis.
2. If  $\mathcal{J}' = \mathcal{I}_j$  and  $\mathcal{I}_i \not\models^r C \sqsubseteq A \in \mathcal{T}$  with  $i > j$ , then  $\rho_{\mathcal{I}_j} \notin C^{\mathcal{I}_j}$ .  
Assume to the contrary that  $\rho_{\mathcal{I}_j} \in C^{\mathcal{I}_j}$ . Since  $\mathcal{J}'$  is obtained from  $\mathcal{I} \times_r \mathcal{J}$  by removing subtrees, we then have  $\rho_{\mathcal{I} \times_r \mathcal{J}} \in C^{\mathcal{I} \times_r \mathcal{J}}$ , thus  $\rho_{\mathcal{J}} \in C^{\mathcal{J}}$ . Since  $\mathcal{I}_j = \mathcal{J}$  before the replacement of  $\mathcal{J}$  with  $\mathcal{J}'$ , we have a contradiction against the induction hypothesis. □

We now turn towards proving Point (ii) above. It is a consequence of Lemma 16 below.

**Lemma 15.** *If  $\mathcal{I}$  is an essential  $\mathcal{T}$ -countermodel, then  $|\Delta^{\mathcal{I}}| \leq |\mathcal{T}|$ .*

*Proof.* Let  $\mathcal{I}$  be an essential  $\mathcal{T}$ -countermodel. Then  $\mathcal{I} \not\models \mathcal{T}$ , but  $\mathcal{I}|_{\text{root}} \models \mathcal{T}$ . It follows that there is a  $C \sqsubseteq A \in \mathcal{T}$  such that  $\rho_{\mathcal{I}} \in C^{\mathcal{I}} \setminus A^{\mathcal{I}}$ . Consequently, there is a simulation  $\sim$  from  $(\mathcal{I}_C, \rho_{\mathcal{I}_C})$  to  $(\mathcal{I}, \rho_{\mathcal{I}})$ . Since  $\mathcal{I}_C$  is tree-shaped, we can assume w.l.o.g. that  $\sim$  is a total function (that is, a homomorphism). To show that  $|\mathcal{I}| \leq |C|$  and thus  $|\mathcal{I}| \leq |\mathcal{T}|$  as required, it clearly suffices to show that  $\sim$  is surjective. Assume that this is not the case, and let  $d \in \Delta^{\mathcal{I}}$  be outside the range of  $\sim$  and  $\mathcal{J} = \mathcal{I}|_{d^{\downarrow}}$ . All descendants of  $d$  must be outside the range of  $\sim$  as well and thus  $\sim$  is a simulation from  $(\mathcal{I}_C, \rho_{\mathcal{I}_C})$  to  $(\mathcal{J}, \rho_{\mathcal{J}})$ . Therefore,  $\rho_{\mathcal{J}} \in C^{\mathcal{J}}$ , which implies  $\mathcal{J} \models C \sqsubseteq A$ , in contradiction to  $\mathcal{I}$  being an essential  $\mathcal{T}$ -countermodel: Property 2 of being essential requires  $\mathcal{J} \models \mathcal{T}$ . □

**Lemma 16.** *Let  $\mathcal{I}_0, \dots, \mathcal{I}_n$  be a sequence of interpretations such that  $\mathcal{I}_{i+1}$  replaces  $\mathcal{I}_i$  in Line 9 for all  $i < n$ . Then  $n \leq |\mathcal{T}| + |\mathcal{T}|^2$ .*

*Proof.* We first show that for every  $i < n$  either

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

For a proof by contradiction assume that there is  $i < n$  such that neither Point 1 nor Point 2 holds. Since  $\mathcal{I}_{i+1}$  is a subinterpretation of some  $\mathcal{I} \times_r \mathcal{I}_i$  and  $\mathcal{I} \times_r \mathcal{I}_i \Rightarrow \mathcal{I}_i$  we obtain that  $\mathcal{I}_{i+1} \Rightarrow \mathcal{I}_i$ . Since  $\mathcal{I}_{i+1}$  is an essential  $\mathcal{T}$ -countermodel, there is a  $C \sqsubseteq A \in \mathcal{T}$  such that  $\mathcal{I}_{i+1} \not\models^r C \sqsubseteq A$ . Let  $\mathcal{J}$  be the subinterpretation of  $\mathcal{I}_i$  determined by the range of the simulation  $\sim$  from  $\mathcal{I}_{i+1}$  to  $\mathcal{I}_i$ . Then  $\rho_{\mathcal{J}} \in C^{\mathcal{J}}$  and so, since  $\rho_{\mathcal{J}} \notin A^{\mathcal{J}}$  because Point 1 does not hold,  $\mathcal{J} \not\models^r C \sqsubseteq A$ .  $\mathcal{I}_i$  is an essential  $\mathcal{T}$ -countermodel and so  $\mathcal{J} = \mathcal{I}$ . But then  $\sim$  is surjective and we have derived a contradiction.

In addition to the property stated above, we also have:

- for all concept names  $A$ , if  $\rho_{\mathcal{I}_i} \in A^{\mathcal{I}_{i+1}}$ , then  $\rho_{\mathcal{I}_{i+1}} \in A^{\mathcal{I}_i}$ ;
- $\mathcal{I}_i \not\Rightarrow \mathcal{I}_{i+1}$ .

By Lemma 15 we have  $|\Delta^{\mathcal{I}_i}| \leq |\mathcal{T}|$  for all  $i \leq n$ . Thus, any convex subsequence  $\mathcal{I}_j, \dots, \mathcal{I}_k$  of the sequence  $\mathcal{I}_0, \dots, \mathcal{I}_n$  such that  $\rho_{\mathcal{I}_i} \in A^{\mathcal{I}_i}$  iff  $\rho_{\mathcal{I}_{i+1}} \in A^{\mathcal{I}_{i+1}}$  holds for all concept names  $A$  and all  $j \leq i < k$  has length bounded by  $|\mathcal{T}|^2$ . □