Proc. the 6th International Workshop on Algorithmic Learning Theory (ALT'95), LNAI 997, Springer-Verlag, 66-79, Oct. 1995

Learning Unions of Tree Patterns Using Queries

Hiroki Arimura, Hiroki Ishizaka, Takeshi Shinohara

Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820, Japan
{arim,ishizaka,shino}@ai.kyutech.ac.jp

Abstract. This paper characterizes the polynomial time learnability of TP^k, the class of collections of at most k first-order terms. A collection in TP^k defines the union of the languages defined by each first-order terms in the set. Unfortunately, the class TP^k is not polynomial time learnable in most of learning frameworks under standard assumptions in computational complexity theory. To overcome this computational hardness, we relax the learning problem by allowing a learning algorithm to make membership queries. We present a polynomial time algorithm that exactly learns every concept in TP^k using $O(kn)$ equivalence and $O(k^2n \cdot \max\{k,n\})$ membership queries, where n is the size of longest counterexample given so far. In the proof, we use a technique of replacing each restricted subset query by several membership queries under some condition on a set of function symbols. As corollaries, we obtain the polynomial time PAC-learnability and the polynomial time predictability of TP^k when membership queries are available. We also show a lower bound $\Omega(kn)$ of the number of queries necessary to learn TP^k using both types of queries. Further, we show that neither types of queries can be eliminated to achieve efficient learning of TP^k. Finally, we apply our results in learning of a class of restricted logic programs, called unit clause programs.

1 Introduction

Inductive learning is a process of finding general concepts from their concrete examples. In this paper, we consider the learnability of the class C^k of unions, which is the collection of unions $c_1 \cup \ldots \cup c_n$ of at most k concepts $c_1, \cdots, c_n$ $(n \leq k)$ in a concept class C. In particular, this paper focuses on the polynomial time learnability of TP^k, the class of unions of at most k tree pattern languages.

A *tree pattern* is a first-order term in formal logic, and the *language defined by a tree pattern* p is the set of all the tree patterns obtained by replacing each variable in p with a tree pattern containing no variable. A set of tree patterns represents the union of the languages of each tree pattern in the set. For a non-negative integer k, TP^k is the class of sets of at most k tree patterns. Although concepts in TP^k are simple, they have characteristics common to a variety of representation frameworks for structured objects such as knowledge representation languages [7, 10], logic programming languages [6, 13], and combinatorial objects like string patterns [1, 3, 5, 9]. Furthermore, computational problems

related to tree patterns are more efficiently solvable than the other representation frameworks; for example, the membership and the containment problems are polynomial time solvable for tree patterns, while the membership problem is NP-complete [1] and the containment problem is undecidable for string patterns [9]. For these reasons, it is a basic but useful practice in machine learning to capture the efficient learnability of TP^k.

Plotkin [16] and Reynolds [17] introduced a *subsumption relation* $\preceq$ over TP^1, and developed a polynomial time algorithm for computing the least upper bound of a given set of tree patterns, called the *lgg*. By computing the lgg of positive examples, TP^1 is polynomial time PAC-learnable, or polynomial time exact learnable using equivalence queries [1].

This subsumption relation $\preceq$ can be extended for a powerset relation $\sqsubseteq$ over TP^k by defining as $P \sqsubseteq Q$ iff $(\forall p \in P)(\exists q \in Q)\ p \preceq q$ for any unions P, Q. Arimura et al. [4] developed a polynomial time algorithm that finds one of the minimal upper bounds (called *mmg*) of a given set of tree patterns with respect to $\sqsubseteq$. Using this algorithm, they showed that the class TP^k is *identifiable in the limit from positive data with consistent and conservative polynomial time update*.

Unfortunately, the convergence time of the algorithm has not been proved to be bounded by any polynomial. The reason is that there may exist several mmg's of the set, while the lgg is unique. In fact, we can prove that for every $k \geq 2$ the class TP^k of unions is not polynomial time learnable in PAC-learning model or exact learning model mentioned above under some assumption in complexity theory, while TP^1 is polynomial time learnable in both models.

One approach to overcome this computational hardness is to relax the problem by allowing a learning algorithm to make *membership queries*. A membership query is any word w, and its answer is "yes" or "no" according to whether w is contained in the target concept. By using membership queries, a learning algorithm can actively collect information on a target concept. Ishizaka *et al.* [8] considered a decision problem closely related to the learnability of TP^k, called the *consistency problem*, and showed that the problem is polynomial time solvable using membership queries for $k = 2$.

In this paper, we present a polynomial time algorithm for learning TP^k using equivalence and membership queries. Under some condition on a set of function symbols, our algorithm LEARN exactly identifies every target concept in TP^k, and outputs "failed" for every target concept not in TP^k. Further, LEARN runs in polynomial time in k and n making $O(kn)$ equivalence queries and $O(k^2 n \cdot \max\{k, n\})$ membership queries, where n is the size of the longest counterexample. Then, the result of Ishizaka *et al.* is generalized by this algorithm for arbitrary positive integer k. Since the algorithm is presented in the paradigm of exact learning by Angluin [2], we can transform in a standard way our learning algorithm into either a *polynomial time PAC-learning algorithm using membership queries* [2], or a *polynomial time prediction algorithm with a polynomial mistake bound using membership queries* [11].

[1] In exact learning of TP^1, we have to assume that there exists the bottom element $\bot$ of TP^1, or that one positive example is initially given to a learning algorithm.

The paper is organized as follows. Section 3 gives a polynomial time learning algorithm using equivalence and restricted subset queries, and shows the correctness and the complexity. Next, Section 4 shows that a subset query can be replaced with several membership queries under some condition on a set of function symbols. Then, we show a polynomial time algorithm for learning TP^k using $O(kn)$ equivalence queries and $O(k^2n \cdot \max\{k, n\})$ membership queries. We also show that any algorithm using equivalence and membership queries requires $\Omega(kn)$ queries. Section 5 shows that neither types of queries can be eliminated to learn TP^k in polynomial time. Section 6 describes an application of the results in Section 3.2 in learning restricted logic programs whose computational mechanisms are only disjunctive definition and unification. Finally, Section 7 concludes the results.

2 Preliminaries

2.1 Unions of tree patterns

Let Σ be a finite alphabet. Each element of Σ is called a *function symbol* and associated with a non-negative integer called an *arity*. A function symbol with arity 0 is also called a *constant*. We assume that Σ contains at least one constant. Let V be a countable set of symbols disjoint from Σ. Each element of V is called a *variable*.

A *first-order term* over Σ is an expression defined recursively as follows: (1) a function with arity 0 or a variable is a first-order term; (2) For a function f with arity n ($n \geq 1$) and first-order terms $t_1, \ldots, t_n$, $f(t_1, \ldots, t_n)$ is a first-order term. Throughout this paper, we will refer to the first-order terms as *tree patterns* by analogy with string patterns [1]. A tree pattern is said to be *ground* if it contains no variable. Tree patterns will normally be denoted by letters p, q, r, and h, and sets of tree patterns will normally be denoted by capital letters P, Q, R, and H, possibly subscripted.

The set of all the tree patterns is denoted by TP and the set of all the ground tree patterns is denoted by $TP(\Sigma)$. For a non-negative integer k, the class of all the sets consisting of at most k tree patterns is denoted by TP^k. Each element in TP^k is called a *union of at most k tree patterns*. Note that TP^0 includes only one element $\emptyset$.

For a set S, $|S|$ denotes the number of elements in S, and 2^S denotes the powerset of S. The *size* of a tree pattern p, denoted by $size(p)$, is the number of symbol occurrences in p minus the number of distinct variables occurring in p. For example, $size(f(g(X, Y), h(X, Z), Y)) = 8 - 3 = 5$. For a set P of tree patterns, $size(P)$ is defined as $\sum_{p \in P} size(p)$. Note that if a tree pattern p contains no variable then $size(p)$ is the total number of symbol occurrences in p.

A *substitution* is a finite set of the form $\{X_1/t_1, \ldots, X_n/t_n\}$, where X_i is a variable, each t_i is a tree pattern different from X_i, and $X_1, \ldots, X_n$ are mutually distinct. An *instance* of a tree pattern p by a substitution $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$, denoted by $p\theta$, is the tree pattern obtained by simultaneously replacing each occurrences of the variable X_i with the term t_i ($1 \leq i \leq n$).

For tree patterns p and q, if there exists a substitution θ such that $p = \theta(q)$, then q is said to be a *generalization* of p, and denoted by $p \preceq q$. If $p \preceq q$ but $q \not\preceq p$, then we define $p \prec q$. If both $p \preceq q$ and $q \preceq p$ hold, then we define $p \equiv q$. In what follows, we do not distinguish any tree patterns which are the same modulo $\equiv$. Now, we extend the relation $\preceq$ for unions. Let $P, Q \in TP^k$. If $(\forall p \in P)(\exists q \in Q)\ p \preceq q$ then P is said to be a *refinement* of Q, or Q is a *generalization* of P, and denoted by $P \sqsubseteq Q$. A refinement P of a union Q is said to be *conservative* if for any $q \in Q$ there is at most one $p \in P$ such that $p \preceq q$. For a tree pattern h and a set H of tree patterns, we write $h \preceq H$ if $\{h\} \sqsubseteq H$ holds.

For a tree pattern p, the *language of* p, denoted by $L(p)$, is the set of all the ground instances of p. More precisely, $L(p) = \{\ w \in TP(\Sigma) \mid w \preceq p\ \}$. For a set P of tree patterns, the *language* of P is also denoted by $L(P)$ and defined as $L(P) = \bigcup_{p \in P} L(p)$. If no confusion arises, we denote also by TP^k the class of languages defined by sets of at most k tree patterns. For example, if Σ is an alphabet $\{tom, bob, mom(\cdot), likes(\cdot,\cdot), \ldots\}$ and P is a union $\{likes(X,X), likes(Y, mom(Y))\}$, the language of P is the following set:

$$L(P) \ = \ \left\{ \begin{array}{l} likes(tom,tom),\ likes(bob,bob),\ likes(mom(tom),mom(tom)),\ \ldots \\ likes(tom,mom(tom)),\ likes(mom(tom),mom(mom(tom))),\ \ldots \end{array} \right\}.$$

For sets P, Q of tree patterns, if $L(P) = L(Q)$ then P is said to be *equivalent* to Q. A set P of tree patterns is said to be *reduced* if there is no $p, q \in P$ $(p \not\equiv q)$ such that $p \preceq q$. For any set $P \in TP^k$ of tree patterns, there exists the unique reduced set $\widehat{P} \in TP^k$ that is equivalent to P. The following property is called the *compactness* of tree pattern languages [4].

Proposition 1. *Suppose that* $|\Sigma| \geq k{+}1$. *Then, for any tree patterns* $p, p_1, \ldots, p_k$, $L(p) \subseteq L(p_1) \cup \cdots \cup L(p_k)$ *if and only if* $p \preceq p_i$ *for some* $1 \leq i \leq k$.

For a nonempty set S of tree patterns, a tree pattern p is said to be a *common generalization* of S if $p \succeq q$ for any q in S. A *least general generalization (lgg)* of S is a common generalization p of S such that $p \preceq q$ for every common generalization q of S. For any set S of tree patterns, a lgg of S always exists, is unique modulo $\equiv$, and is polynomial time computable [16, 17]. For tree patterns p, q, we denote by $p \sqcup q$ the lgg of the pair $\{p, q\}$. Furthermore, the following properties hold (see e.g. [16, 17]).

Proposition 2. *Let* p, q *be tree patterns,* w *be a ground tree patterns, and* $H, H' \in TP^k$ *be finite sets of tree patterns. Suppose that* $|\Sigma| > 1$. *Then the following propositions hold.*

1. *If* $p \preceq q$ *then* $size(p) \geq size(q)$, *and if* $p \prec q$ *then* $size(p) > size(q)$.
2. $p \succeq q$ *if and only if* $L(p) \supseteq L(q)$.
3. $H \sqsubseteq H'$ *if and only if* $L(H) \subseteq L(H')$ *when* $|\Sigma| > k$.
4. $w \preceq H$ *if and only if* $w \in L(H)$.
5. *If* $H \sqsubseteq H'$ *and* $h \preceq H'$ *then* $H \cup \{h\} \sqsubseteq H'$

2.2 The learning problem

We employ the standard protocol of exact learning from equivalence and membership queries [2]. Let $H_* \in TP^k$ be the *target* union to be identified. A *learning algorithm* $\mathcal{A}$ may collect information about H_* using equivalence and membership queries. An *equivalence query* is to propose any hypothesis H in TP^k. If $L(H) = L(H_*)$ then the answer to the query is "yes," and $\mathcal{A}$ has succeeded in the inference task. Otherwise, the answer is "no," and $\mathcal{A}$ receives any ground tree pattern w in the symmetric difference $L(H) \oplus L(H_*)$ as a *counterexample*. A counter example w is said to be *positive* if $w \in L(H_*)$ and *negative* if $w \in L(H)$. A *membership query* is to propose a ground tree pattern w. The answer to the membership query is "yes" if $w \in L(H_*)$, and "no" otherwise. An equivalence and a membership queries are denoted by $EQUIV(H)$ and $MEMB(w)$, respectively. The goal of a learning algorithm $\mathcal{A}$ is *exact identification* in polynomial time, that is, $\mathcal{A}$ must halt and output a union $H \in TP^k$ that is equivalent to H_*. Furthermore at any stage in learning, the running time of $\mathcal{A}$ must be bounded by a polynomial in the size of H_* and the size of the longest counterexample returned by equivalence queries so far.

Because the problem of determining whether two unions in TP^k are equivalent is solvable in polynomial time [10], both types of queries are efficiently computable by a teacher.

3 Learning Unions of Tree Patterns Using Queries

In this section, we first prove that there exists a polynomial time algorithm that correctly identifies any union in TP^k using equivalence and restricted subset queries. Then in the next section, we prove that every restricted subset query made in LEARN can be replaced with several membership queries. This yields a modified version of the algorithm that exactly learns every union in TP^k in polynomial time using equivalence and membership queries.

3.1 The learning algorithm

Figure 1 gives our learning algorithm LEARN, which uses equivalence and restricted subset queries to learn TP^k. A *restricted subset query*, denoted by $SUBSET(H)$, is to propose any hypothesis H in TP^k, and the answer is either "yes" or "no" according to whether $L(H) \subseteq L(H_*)$. The algorithm starts with the most specific hypothesis $\emptyset$, and searches hypotheses space TP^k from more specific to more general. The algorithm carefully generalizes each element of a hypothesis by making restricted subset queries so that only positive counterexamples are provided.

We give an example to illustrate the operation of the algorithm. Suppose that the alphabet is $\Sigma = \{ cat, dog, beef, pork, orange, banana, h(\cdot), f(\cdot), eat(\cdot,\cdot) \}$ and the target concept is $H_* = \{ eat(X, m(Y)), eat(h(X), Y) \}$, where h and m are abbreviations of *hungry* and *meat*, respectively. The following table shows

the computation of LEARN. In each row, the tree pattern underlined in the fourth column denotes the updated element in H_n. At each stage $n \geq 1$ LEARN asks an equivalence query with a hypothesis H_{n-1}, receives a counterexample w_n if H_{n-1} is not equivalent to the target, and updates the hypothesis H_n by either replacing some element $h \in H_{n-1}$ with the lgg $h \sqcup w_n$ (Gen), or simply adding w_n into H_{n-1} (Add). At stage 6, LEARN receives "yes" as the answer to the equivalence query, and then succeeds in learning H_*.

n	counterexample w_i	update	hypothesis H_n
0	—	—	$\{\ \}$
1	$eat(cat, m(beef))$	Add	$\{\underline{eat(cat, m(beef))}\}$
2	$eat(dog, m(beef))$	Gen	$\{\underline{eat(X, m(beef))}\}$
3	$eat(h(cat), orange)$	Add	$\{eat(X, m(beef)), \underline{eat(h(cat), orange)}\}$
4	$eat(h(dog), m(pork))$	Gen	$\{\underline{eat(X, m(Y))}, eat(h(cat), orange)\}$
5	$eat(h(dog), banana))$	Gen	$\{eat(X, m(Y)), \underline{eat(h(X), Y)}\}$
6	*"yes"*	—	—

3.2 The correctness and the complexity of the learning algorithm

Let Σ be the alphabet such that $|\Sigma| > k$, and $H_* \in TP^k$ be the target union. In what follows, let $H_0, H_1, \ldots, H_i, \ldots$ and $w_1, w_2, \ldots, w_i, \ldots$ $(i \geq 0)$, respectively, be the sequence of hypotheses asked in the equivalence queries by LEARN and the sequence of counterexamples returned by the queries. H_0 is the initial hypothesis $\emptyset$, and at each stage $i \geq 1$, LEARN makes an equivalence query $EQUIV(H_{i-1})$, receives a counterexample w_i to the query, and produce a new hypothesis H_i from w_i and H_{i-1}.

Lemma 3. *For each $i \geq 0$, w_i is a positive counterexample and H_i is a refinement of H_*.*

Proof. The proof is by induction on the number of iterations $i \geq 0$ of the main loop. If $i = 0$ then the result immediately follows since the hypothesis $H_0 = \emptyset$ defines the smallest language $\emptyset$.

Assume inductively that the result holds for any number of iteration of the main loop less than i, and H_i is defined. Since $H_{i-1} \sqsubseteq H_*$ by induction hypothesis, and since w_i is a counterexample witnessing $L(H_{i-1}) \neq L(H_*)$, we know $L(H_{i-1}) \subset L(H_*)$ by Proposition 2. Thus, w_i must be positive.

On the other hand, H_i is obtained from both of H_{i-1} and w_i at Line 5 of Figure 1. There are two cases for the construction of H_i. (i) If the answer to the query $SUBSET(h \sqcup w_i)$ is "yes" for some $h \in H_i$, then $h \sqcup w_i \preceq H_*$ by (3) of Proposition 2. By induction hypothesis, we know $H_{i-1} \sqsubseteq H_*$. Thus, we have $H_i = (H_{i-1} - \{h\}) \cup \{h \sqcup w_i\} \sqsubseteq H_*$ by (5) of Proposition 2. (ii) Otherwise,

Procedure: $LEARN$
Given: the equivalence and the subset oracles for the target set $H_* \in TP^k$.
Output: a set H of at most k tree patterns equivalent to H_*.
begin
1 $H := \emptyset$;
2 **until** $EQUIV(H)$ returns "yes" **do**
3 **begin**
4 let w be a counterexample returned by the equivalence query;
5 **if** there is some $h \in H$ such that $SUBSET(h \sqcup w)$ returns "yes" **then**
6 generalize H by replacing h with $h \sqcup w$
7 **else if** $|H| < k$ **then**
8 generalize H by adding w into H
9 **else**
10 return "failed"
11 **endif**
12 **end** /* main loop */
13 return H;
end

Fig. 1. A learning algorithm for TP^k using equivalence and restricted subset queries.

$H_{i-1} \sqsubseteq H_*$ by induction hypothesis. Since $w_i \in L(H_*)$ holds from the induction hypothesis, we have $w_i \preceq H_*$ by (4) of Proposition 2. Thus, (5) of Proposition 2 shows that $H_i = H_{i-1} \cup \{w_i\} \sqsubseteq H_*$. □

The next lemma states that the number of tree patterns in a produced hypothesis H_i does not diverge in the learning process. This makes it possible for LEARN to reject every target concept that does not belong to TP^k.

Lemma 4. *For each $i \geq 0$, H_i is a conservative refinement of H_*.*

Proof. If $i = 0$ then $H_0 = \emptyset$ is obviously a conservative refinement of H_*. Assume inductively that H_{i-1} is a conservative refinement of H_*, and H_i is defined. By Lemma 3, we know that H_i is a refinement of H_*. There are two cases for the construction of H_i. (i) If the answer to the query $SUBSET(h \sqcup w_i)$ at Line 5 of Figure 1 is "yes" for some $h \in H_{i-1}$, then $h \sqcup w_i \preceq h_*$ for some $h_* \in H_*$ by (3) of Proposition 2. Since h is the unique tree pattern in H_{i-1} satisfying $h \preceq h_*$ from the conservativeness of H_{i-1}, $h \sqcup w_i$ is also the unique tree pattern in H_i satisfying $h \sqcup w_i \preceq h_*$. (ii) Otherwise, there is no tree pattern h in H_{i-1} satisfying $h \sqcup w_i \preceq h_*$ for some $h_* \in H_*$. Therefore, w_i is the only tree pattern h in H_i satisfying $h \preceq h_*$ because w_i is a positive counterexample and $H_i = H_{i-1} \cup \{w_i\}$. Combining (i) and (ii) above, we prove the result. □

Corollary 5. *For each $i \geq 0$, $H_i \in TP^k$.*

To prove the termination of the algorithm, we need the following technical lemma. Let $\mathbf{N}_\infty = \mathbf{N} \cup \{\infty\}$. For k-tuples of elements in $\mathbf{N}_\infty$, we define a strict partial order on k-tuples as $\langle x_1, \ldots, x_k \rangle > \langle y_1, \ldots, y_k \rangle$ iff $x_i \geq y_i$ for all i and $x_i > y_i$ for some i.

Lemma 6. *Let $Z_0 > Z_1 > \cdots > Z_i > \cdots (Z_i \in (N_\infty)^k, i \geq 0)$ be the properly decreasing sequence of k-tuples of elements in $\mathbf{N}_\infty$. If n is the maximum finite integer appearing in the sequence then the length of the sequence is at most $k(n+2)$.*

Theorem 7. *Let k be any positive integer and $|\Sigma| > k$. Then, the algorithm LEARN exactly identifies every set H_* of at most k tree patterns in time polynomial in k and n making $O(kn)$ equivalence queries and $O(k^2 n)$ restricted subset queries, where n is the size of the longest counterexample seen so far.*

Proof. By the construction of the algorithm, if the algorithm terminates then the last hypothesis H_i is equivalent to the target H_*. Further from Corollary 5, we know that all the hypotheses produced so far are members in TP^k. Therefore, it is sufficient to show the termination in polynomial time.

Let $\perp$ be a special tree pattern such that $\perp \prec p$ for any tree pattern p and $size(\perp) = \infty$. Suppose H_* is reduced and has k elements. For each $n \geq 0$, we associate with H_n a k-vector $\widetilde{H}_n$ over $TP \cup \{\perp\}$ as follows. First, for $H_* = \{h_1^*, \ldots, h_k^*\}$, let $\widetilde{H}_*$ be any k-vector $\langle h_1^*, \ldots, h_k^* \rangle$. For each H_n $(n \geq 0)$, let $\widetilde{H}_n$ be a vector $\langle h_1, \ldots, h_k \rangle$ such that $h_i \in H_n$ and $h_i \preceq h_i^*$ for each $1 \leq i \leq k$. We put $h_i = \perp$ if there is no $h \in H_n$ satisfying $h \not\preceq h_i^*$. Since each H_n is a conservative refinement of H_*, $\widetilde{H}_n$ is uniquely determined.

Let H_n be defined, and let $H_{n-1} = \langle p_1, \ldots, p_k \rangle$ and $H_n = \langle q_1, \ldots, q_k \rangle$. By construction, there is exactly one $1 \leq i \leq k$ such that $p_i \neq q_i$ and $p_j = q_j$ for all $j \neq i$. Then, we show $p_i \prec q_i$ as follows. (i) The case where H_n is obtained from H_{n-1} by replacing p_i with $p_i \sqcup w_n$ at Line 6 of Figure 1. From the proof of Lemma 4, we observe that $p_i \sqcup w_n$ is the unique member h of H_n such that $h \preceq h_i^*$. Thus, q_i must be $p_i \sqcup w_n$, and $p_i \preceq (p_i \sqcup w_n)$. Since w_n is a positive counterexample, we have $w_n \not\preceq p_i$. Hence, $p_i \prec (p_i \sqcup w_n) = q_i$.

(ii) The case where H_n is obtained from H_{n-1} by adding w_n into H_{n-1} at Line 8 of Figure 1. Since queries $SUBSET(h \sqcup w_n)$ returns "no" for all $h \in H_{n-1}$, we can see that p_i must be $\perp$. Thus $p_i = \perp \prec w_n = q_i$.

Consider the sequence $Z_0, Z_1, \ldots, Z_n, \ldots$ of k-vectors over $\mathbf{N}_\infty$, where $\widetilde{H}_n = \langle h_1^n, \ldots, h_k^n \rangle$ and $Z_n = \langle size(h_1^n), \ldots, size(h_k^n) \rangle$ for all $n \geq 0$. Combining (i) and (ii) above, for each $n \geq 0$ we have $h_i^{n-1} \preceq h_i^n$ for all i and $h_i^{n-1} \prec h_i^n$ for some i. From (1) of Proposition 2, we know the sequence $Z_0, Z_1, \ldots, Z_n, \ldots$ satisfies the condition of Lemma 6. Thus, the length of the sequence is at most $k(n+2)$, where n is the size of the longest counterexample given so far. Since the lgg is polynomial time computable, the computation time of the algorithm is clearly bounded by a polynomial in k and n. The numbers of equivalence and restricted subset queries are bounded by $k(n+2)$ and $k^2(n+2)$, respectively. This completes the proof. □

4 Replacing a Subset Query with Membership Queries

In this section, we show that a modified version of our learning algorithm which uses several membership queries instead of a restricted subset query also works well under some condition for Σ.

Lemma 8. *Let k be a positive integer. Suppose that $|\Sigma| > k$ and that Σ contains at least $k-1$ function symbols of nonzero arity. Then, for any tree pattern r with n variables there exists a finite set $G(r)$ of $k+n-1$ ground instances of r satisfying the following: for any tree patterns $p_1, \ldots, p_l$ $(l \leq k)$, $G(r) \subseteq L(p_1) \cup \cdots \cup L(p_l)$ if and only if $r \preceq p_i$ for some $1 \leq i \leq l$.*

Proof. The if part of the statement is trivial. Thus, we show the only-if part. Without loss of generality, we may assume that Σ contains $k-1$ unary function symbols $f_1, \ldots, f_{k-1}$ and two constants a, b. For function symbols of arity more than one, we can fill the extra places by some dummy symbol, say a. If r is ground then the proof is done. Thus, we assume that r contains n variables $x_1, \ldots, x_n$. Let $\sigma_1, \ldots, \sigma_{k-1}$ and $\theta_1, \ldots, \theta_n$ be the substitutions that replace each variable by some ground tree pattern as defined in the followings. For each $i = 1, \ldots, k-1$ and for each $j = 1, \ldots, n$, let

$$\sigma_i(x_j) = \overbrace{f_i(f_i(\cdots f_i}^{j}(a)\cdots)).$$

For each $i = 1, \ldots, n$ and for each $j = 1, \ldots, n$, let

$$\theta_i(x_j) = \begin{cases} a & \text{, if } i = j, \\ b & \text{, otherwise.} \end{cases}$$

Let $G(r) = \{\sigma_1(r), \ldots, \sigma_{k-1}(r), \theta_1(r), \ldots, \theta_n(r)\}$. Suppose to the contrary that $r \not\preceq p_i$ for any $1 \leq i \leq l$.

Consider the case where $l = k$. For any $1 \leq i \leq k$, if $L(p_i)$ contains $\sigma_{j_1}(r)$ and $\sigma_{j_2}(r)$ for distinct j_1, j_2, then $r \preceq p_i$ holds. This is the contradiction. Thus, we needs at least $k-1$ distinct tree patterns to contain all of ground tree patterns $\sigma_1(r), \ldots, \sigma_{k-1}(r)$. Without loss of generality, assume that $\sigma_1(r) \in L(p_1), \ldots, \sigma_{k-1}(r) \in L(p_{k-1})$. If $L(p_i)$ contains at least one $\theta_j(r)$ for some $1 \leq j \leq n$, then $r \preceq p_i$ immediately holds, and the contradiction also follows. Thus, $\theta_j(r)$ is contained by none of $L(p_1), \ldots, L(p_{k-1})$ for any $1 \leq j \leq n$. This implies that $L(p_k)$ must contain all of ground tree patterns $\theta_1(r), \ldots, \theta_n(r)$. Since the least common generalization of $\theta_1(r), \ldots, \theta_n(r)$ is r, we have $r \preceq p_k$. This derives the contradiction. Hence, $r \preceq p_i$ for some $1 \leq i \leq k$. A similar discussion can prove the result for the cases where $l < k$. □

By the lemma shown above, it is easily seen that $L(r) \subseteq L(H_*)$ if and only if $w \in L(H_*)$ for all $w \in G(r)$. Therefore, we can replace each restricted subset query made in the learning algorithm LEARN by at most $k+n-1$ membership queries, where n is the size of the maximum pattern in the current hypothesis H (Figure 2). Since n is bounded by the size of the longest counterexample seen so far, we have the following theorem from Theorem 7 in the previous section.

Procedure: $SUBSET(p)$
Input: a tree pattern p.
Given: the membership oracle for the target set H_*.
Output: "yes" if $L(p)$ is a subset of $L(H_*)$, and "no" otherwise.
begin
  for each $r \in G(p)$ **do** query $MEMB(r)$;
  if all the answers returned by the membership queries are "yes" **then**
    return "yes"
  else
    return "no"
  endif
end

Fig. 2. A procedure for replacing a subset query with several membership queries

Theorem 9. *Suppose that $|\Sigma| > k$ and that Σ contains at least $k-1$ function symbols of nonzero arity. For each positive integer k, the algorithm LEARN exactly identifies every set H_* of at most k tree patterns in time polynomial in k and n making $O(kn)$ equivalence queries and $O(k^2 n \cdot \max\{k, n\})$ membership queries, where n is the size of the longest counterexample seen so far.*

Corollary 10. *Let k be any positive integer. Suppose that $|\Sigma| > k$ and that Σ contains at least $k-1$ function symbols of nonzero arity. Then, the following statements hold.*

- *There exists a polynomial time PAC-learning algorithm for TP^k using membership queries. (See Angluin* [2] *for definitions.)*
- *There exists a polynomial time prediction algorithm for TP^k with a polynomial mistake bound using membership queries (excluding the element to be predicted). (See Littlestone* [11] *for definitions.)*
- *There exists a polynomial time algorithm that solves the consistency problem for TP^k using membership queries, given a minimally consistent membership oracle. (See Ishizaka et al.* [8] *for definitions.)*

Proof. For proofs of (a) and (b), see Angluin [2] and Littlestone [11]. The transformations describes in these literatures also works when membership queries are allowed. To prove (c), we describe an algorithm $\mathcal{C}$ that solves the consistency problem for TP^k working as an environment of the learning algorithm LEARN. $\mathcal{C}$ receives sets Pos and Neg of positive and negative examples, and answers queries asked by LEARN as follows. Let $H_* \in TP^l (l \geq 0)$ be a hypothesis of the smallest cardinality that is consistent with Pos and Neg. For an equivalence query with H, the answer to LEARN is "yes" if H is consistent with Pos and Neg. Otherwise, the answer to LEARN is "no," and $\mathcal{C}$ returns any elements w in

$Pos - L(H)$ or $Neg \cap L(H)$ as a counterexample. For a membership query with w, $\mathcal{C}$ asks the *minimally consistent membership oracle* [8] whether $w \in L(H_*)$, and returns this answer to LEARN. Since $\mathcal{C}$ correctly answers any queries made by LEARN as though H_* is the target hypothesis, if H_* belongs to TP^k then LEARN eventually finds a hypothesis consistent with Pos and Neg, and at worst LEARN may find H_*. Otherwise, LEARN returns "failed." It is easy to observe that the time complexity of $\mathcal{C}$ is bounded by the total size of examples in Pos and Neg. □

We also have the following lower bound from a general lower bound result of Maass and Turán [12].

Theorem 11. *Any algorithm that exactly identifies all the unions of at most k tree patterns using equivalence and membership queries must make $\Omega(kn)$ queries in the worst case, where n is the size of the longest counterexample seen so far.*

Proof. We say a concept class C *shatters* a set $U \subseteq \Sigma^*$ if $\{U \cap c \mid c \in C\} = 2^U$ holds. The *VC-dimension* of C is the cardinality of the largest set $U \subseteq \Sigma^*$ that is shattered by C. Let TP_n^k be the subclass of TP^k for which all tree patterns are of size at most n. We first show that the the VC-dimension of TP_n^k is $\Omega(kn)$. Suppose that Σ contains at least two constants $0, 1$, and k binary function symbols $f^{(1)}, \ldots, f^{(k)}$. Let f be any binary symbol, and let $S = \{w_1, \ldots, w_n\}$ be the set of tree patterns over $\{0, 1, f\}$, where w_i is a tree pattern in TP_{2n-1}

$$w_i = \overbrace{f(b_1, f(\cdots f}^{n}(b_{n-1}, b_n)\cdots)),$$

such that b_j is 0 if $j = i$ and 1 otherwise. Elements of S correspond with bit-vector representations of the elements of the set $\{1, \ldots, n\}$. Consider tree patterns in TP_{2n-1} of the form

$$\overbrace{f(t_1, f(\cdots f}^{n}(t_{n-1}, t_n)\cdots)), \qquad (t_1 \cdots t_{n-1} t_n \in \{x, 0\}^n),$$

which correspond with the bit-vector representations of all the subsets of $\{1, \ldots, n\}$. Then, we can observe that for any subset $T \subseteq S$, there is a tree pattern p of this form such that $T = L(p) \cap S$. Thus, we know TP_{2n-1} shatters a set S of cardinality n.

We generalize this construction for TP_{2n-1}^k. Let $S_1, \ldots, S_k$ be mutually disjoint sets of tree patterns such that S_l is the set obtained from S by replacing each occurrence of f with $f^{(l)}$ $(1 \leq l \leq k)$, and let U be the direct sum $S_1 \cup \cdots \cup S_k$ of cardinality kn. Then, a similar argument shows that TP_{2n-1}^k shatters the set U, and thus the VC-dimension of TP_n^k is $\Omega(kn)$.

Maass and Turán [12] show that any algorithm that exactly learns a concept class C using equivalence and membership queries must make $\Omega(d)$ queries even when arbitrary hypotheses are allowed, where d is the VC-dimension of C. Further, the adversary described in [12] can be modified to choose counterexamples only from U. Since U contains only tree patterns of size at most $2n - 1$, the total number of queries are bounded below by $\Omega(kn)$, where n is the size of the longest counterexample given so far. □

5 Insufficiency of Learning with Membership Queries Alone

In this section, we show that membership queries alone are insufficient for learning of TP^k. Since Ishizaka *et al.* [8] has shown that the problem of finding a union in TP^k consistent with given examples is NP-complete, we know from [14] that equivalence queries alone are insufficient in exact learning model assuming $P \neq NP$, and that random examples alone are insufficient in PAC-learning model assuming $RP \neq NP$. Combining these results and the following theorem, neither equivalence nor membership queries can be eliminated to learn TP^k.

Theorem 12. *Any algorithm that exactly identifies all the unions of at most k tree patterns using membership queries alone must make $\Omega(2^n)$ queries in the worst case, where n is the longest counterexample, even when Σ contains one binary symbol and two constants.*

Proof. Suppose that Σ contains at least two constants $0, 1$, and one binary symbol f. Consider tree patterns over Σ that have the form

$$\overbrace{f(b_1, f(\cdots f}^{n}(b_{n-1}, b_n)\cdots)), \qquad (b_1 \cdots b_{n-1} b_n \in \{0,1\}^n)$$

which correspond with all binary sequences of length $n \geq 1$. There are 2^n such tree patterns, and the intersection of the languages of two tree patterns is empty if they are distinct. Thus by [2], we show that any algorithm using membership queries alone requires at least $2^n - 1$ queries in the worst case. □

6 Application to Learning from Entailment

In this section, we describe an application of the results in Section 3 to learning from entailment. *Learning from entailment* is exact learning suitable for learning logic programs. In learning from entailment [7, 13], a hypothesis is any logic program H and an example is any clause F in the same class. An equivalence test "$L(G) = L(H)$?" and a membership test "$F \in L(H)$?" are replaced by a logical equivalence "$G \Leftrightarrow H$?" and an entailment "$H \models F$?", respectively. Exact learning and PAC-learning based on entailment are defined similarly as usual one.

We consider a class of a restricted logic programs that consists of only unit clauses. We call the class *unit clause programs* and denote by UCP. The predictability and the learnability of general DNF has been a long-standing open problem in computational learning theory. It is easy to see that a log-space transformation from the consistency problem for k-term DNF to that for TP^k given in [8] is actually a prediction preserving reduction of Pitt and Warmuth [15] from DNF to $\bigcup_{k\geq 0} TP^k$. Thus, we have the following theorem.

Theorem 13. *If UCP are polynomial time predictable with polynomial mistake bound with respect to entailment then so are general DNF, even when Σ contains only two binary symbols and one constant.*

Since each clause is unit, we can characterize the entailment $\models$ by the subsumption $\sqsupseteq$ as follows; $\forall(A_1)\& \cdots \&\forall(A_n) \models \forall(B_1)\& \cdots \&\forall(B_n)$ if and only if $\{A_1, \ldots, A_n\} \sqsupseteq \{B_1, \ldots, B_n\}$. Since the running time of the algorithm LEARN is a polynomial in k and n, the next theorem immediately follows from Theorem 7.

Theorem 14. *Let Π be any set of predicate symbols and Σ any set of function symbols. Then, there exists an algorithm that exactly identifies every program H_* in UCP in time polynomial in the size of H_* as a string and the size of the maximum counterexample seen so far, using equivalence and membership queries with respect to entailment.*

These results indicate that membership queries are necessary for learning even very restricted logic programs whose computational mechanisms are only disjunctive definition and unification.

7 Concluding Remarks

In this paper, we investigated efficient learning of TP^k, and presented a polynomial time learning algorithm using equivalence and membership queries. We also showed several hardness results, which suggests that membership queries are necessary for efficiently learning TP^k, in addition to equivalence queries or random examples.

In Section 5, we proved that TP^k is neither polynomial time exact-learnable nor polynomial time PAC-learnable in terms of TP^k when membership queries are not available. Note, however, that this negative result does not eliminate the possibility of a polynomial time learning algorithm in terms of a larger hypothesis space than TP^k, or a polynomial time prediction algorithm with polynomial mistake bound. Hence, it is a future problem to investigate the polynomial time predictability of TP^k in mistake bound model without membership queries.

For some subclasses PAT of string pattern languages, a minimal upper bounds of a finite set is efficiently computable in terms of PAT^k [5]. However, the convergence of the algorithm is not shown to be polynomial. Thus, it is another future problem to generalize the result of the paper for unions of string patterns.

Constrained atoms are nonrecursive definite Horn clauses with a subterm property, and the class of sets of constrained atoms includes the class UCP. Page Jr. independently showed in his PhD. thesis an extension of our Theorem 14 in Section 6 (personal communication, June 1995). He showed that the class of finite sets of constrained atoms is efficiently learnable using equivalence and membership queries in the framework of learning from entailment. However, it is still open whether we can use the techniques developed in Section 3 to achieve efficient learning of sets of constrained atoms without entailment queries.

Acknowledgements

The first author would like to thank Steffen Lange, David Page Jr., Sin-ichi Shimozono, and Kouichi Hirata for discussions and suggestions on this issue. He also thanks the anonymous referees for their valuable comments to improve this paper.

References

1. D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
2. D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.
3. S. Arikawa, S. Kuhara, S. Miyano, A. Shinohara, and T. Shinohara. A learning algorithm for elementary formal systems and its experiments on identification of transmembrane domains. In *Proc. the 25th HICSS*, 675–684, 1992.
4. H. Arimura, T. Shinohara, and S. Otsuki. A polynomial time algorithm for finding finite unions of tree pattern languages. In *Proc. the 2nd NIL*, LNAI 659, Springer-Verlag, 118–131, 1991.
5. H. Arimura, T. Shinohara, and S. Otsuki. Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data. In *Proc. the 11th STACS*, LNCS 775, Springer-Verlag, 649–660, 1994.
6. H. Arimura, H. Ishizaka, T. Shinohara, and S. Otsuki. A generalization of the least general generalization. *Machine Intelligence*, 13, 59–85, Oxford Univ. Press, 1994.
7. M. Frazier and L. Pitt. CLASSIC Learning. In *Proc. COLT'94*, 23–34, 1994.
8. H. Ishizaka, H. Arimura, and T. Shinohara. Finding tree patterns consistent with positive and negative examples using queries In *Proc. ALT'94*, LNAI872, 317–332, 1994.
9. T. Jiang, A. Salomaa, K. Salomaa, and S. Yu. Inclusion is undecidable for pattern languages. In *Proc. 20th ICALP*, pp. 301–312, LNCS 700, Springer, 1993.
10. G. Kuper, K. McAloon, K. Palem, K. Perry. Efficient Parallel Algorithm for Anti-Unification and Relative Complement. In *Proc. the 3rd LICS*, 111–120, 1988.
11. N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2, 285–318, 1988.
12. W. Maass and G. Turán. Lower bound methods and separation results for on-line learning models. Machine Learning, 9, 107–145, 1992.
13. C. D. Page Jr. and A. M. Frisch. Generalization and Learnability: A Study of Constrained Atoms. S. Muggleton, editor, *Inductive Logic Programming*, 29–61, 1992.
14. L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *JACM*, 35:965–984, 1988.
15. L. Pitt and M. K. Warmuth. Prediction preserving reduction. JCSS, 41:430–467, 1990.
16. G. D. Plotkin. A note on inductive generalization. In *Machine Intelligence 5*, 153–163. Edinburgh University Press, 1970.
17. J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. In *Machine Intelligence 5*, 135–152. Edinburgh University Press, 1970.

This article was processed using the LaTeX macro package with LLNCS style