Proc. the 11th Symposium on Theoretical Aspect of Computer Science (STACS'94),
LNCS 775, Springer-Verlag, 649-660, Feb. 1994

Finding Tree Patterns Consistent with Positive and Negative Examples Using Queries

Hiroki Ishizaka Hiroki Arimura Takeshi Shinohara
{ishizaka, arim, shino}@ai.kyutech.ac.jp

Department of Artificial Intelligence Kyushu Institute of Technology
680-4, Kawazu, Iizuka 820, Japan
TEL: 0948-29-7638, FAX: 0948-29-7601

Abstract. This paper is concerned with the problem of finding a hypothesis in $\mathcal{TP}^2$ consistent with given positive and negative examples. The hypothesis class $\mathcal{TP}^2$ consists of all the sets of at most two tree patterns and represents the class of unions of at most two tree pattern languages. Especially, we consider the problem from the point of view of the consistency problem for $\mathcal{TP}^2$. The consistency problem is a problem to decide whether there exists a consistent hypothesis with given positive and negative data within some fixed hypothesis space. Efficient solvability of that problem is closely related to the possibility of efficient machine learning or machine discovery. Unfortunately, however, the consistency problem is known to be NP-complete for many hypothesis spaces, including the class $\mathcal{TP}^2$. In order to overcome this computational hardness, in this paper, we try to use additional information obtained by making queries. First, we give an algorithm that, using restricted subset queries, solves the consistency problem for $\mathcal{TP}^2$ in time polynomial in the total size of given positive and negative examples. Next, we show that each subset query made by the algorithm can be replaced by several membership queries under some condition on a set of function symbols. As a result, we have that the consistency problem for $\mathcal{TP}^2$ is solved in polynomial time using membership queries.

1 Introduction

The consistency problem is a problem to decide whether there exists a consistent hypothesis with given positive and negative data within some fixed hypothesis space. A related search problem, called a fitting, to find such a consistent hypothesis is very essential in machine learning. For example, in the context of PAC learning [11], polynomial time computability of a fitting for a class of polynomial Vapnik-Chervonenkis dimension is sufficient for the class to be polynomial time learnable [7].

Unfortunately, the consistency problem is shown to be NP-complete for many hypothesis spaces such as k-term DNF [8] and regular patterns [5]. To realize efficient learning algorithms, we have to overcome this computational hardness. In most studies on practical machine learning, a target hypothesis space is restricted so that the consistency problem becomes efficiently solvable. Another approach is to use information in addition to initially given positive and negative examples. In this paper, we adopt the latter approach, that is, we investigate an efficient algorithm for solving a consistency problem *using queries*.

A tree pattern is a first order term. The language of a tree pattern p is the set of all the ground instances of p. The tree pattern p is consistent with given positive and negative examples if and only if the language of p includes all the positive examples and no negative example. The consistency problem for the class $\mathcal{TP}$ of tree patterns is solvable in polynomial time, because the least general generalization [10, 9] of positive examples is always consistent with both positive and negative examples whenever there exists a tree pattern consistent with the both examples.

In this paper, we focus on the consistency problem for the hypothesis space $\mathcal{TP}^2$, where $\mathcal{TP}^k$ consists of all the sets of at most k tree patterns and represents the class of unions of at most k tree pattern languages. The consistency problem for $\mathcal{TP}^k$ is also known to be NP-complete even if $k = 2$. We consider two kinds of queries which the given algorithm is allowed to use. The one is a restricted subset query and the other is a membership query. It is well-known that the inclusion and the membership are undecidable and NP-complete respectively for starndard string pattern languages, while both of them are solvable in polynomial time for tree pattern languages. We assume a teacher, called a minimally consistent oracle, that answers queries from the algorithm. A minimally consistent oracle answers each query according to some fixed hypothesis H in $\mathcal{TP}^k$ such that H is consistent with given positive and negative examples and the number of elements of H is less than or equal to that of any consistent hypotheses. First, we give an algorithm that, using restricted subset queries, solves the consistency problem for $\mathcal{TP}^2$ in time polynomial in the total size of given positive and negative examples. Next, we show that each subset query made by the algorithm can be replaced by several membership queries under some condition on a set of function symbols. As a result, we prove that the consistency problem for $\mathcal{TP}^2$ is solved in polynomial time using membership queries.

2 Preliminaries

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*. We adopt some informal notational conventions for the symbols. Variable symbols will normally be denoted by X, Y, and Z, possibly subscripted. Constant symbols will normally be denoted by a, b, and c. Other function symbols will normally be denoted by f, g, and h.

A *tree pattern* over Σ and V is a first order term defined recursively as follows:

1. A function with arity 0 or a variable is a tree pattern.
2. For a function f with arity n $(n \geq 1)$ and tree patterns $t_1, \ldots, t_n$, $f(t_1, \ldots, t_n)$ is a tree pattern.

For a non-variable tree pattern t, the outer-most function symbol of t is called the *principal functor* of t.

For a set S, $|S|$ denotes the number of elements in 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 S of tree patterns, $size(S)$ is defined as $\sum_{p \in S} size(p)$. Note that if a tree pattern p contains no variable then $size(p)$ is the total number of symbol occurrences in p.

A tree pattern is said to be *ground* if it contains no variable. The set of all the tree patterns is denoted by $\mathcal{TP}$ and the set of all the ground tree patterns is denoted by $\mathcal{TP}(\Sigma)$. For some non-negative integer k, the class of all the sets consisting of at most k tree patterns is denoted by $\mathcal{TP}^k$ and called a *hypothesis space*. Each element in $\mathcal{TP}^k$ is called a *hypothesis*. As a convention, we define $\mathcal{TP}^0 = \emptyset$ (empty set).

A *substitution* θ 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 the variables $X_1, \ldots, X_n$ 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 from p by simultaneously replacing each occurrence of the variable X_i by the term t_i $(1 \leq i \leq n)$.

For a tree pattern p, the *language of* p, denoted by $L(p)$, is the set of all the ground instances of p. For a set S of tree patterns, the *language of* S is also denoted by $L(S)$ and defined as $L(S) = \bigcup_{p \in S} L(p)$. For a set T of ground tree patterns, a set S of tree patterns is said to be *reduced* with respect to T if $T \subseteq L(S)$ and $T \not\subseteq L(S - \{p\})$ for any $p \in S$.

If a tree pattern q is an instance of p, that is, there exists a substitution θ such that $p\theta = q$, then p is said to be a *generalization* of q and denoted by $p \succeq q$. If $p \succeq q$ but $q \not\succeq p$, then p is said to be a *proper generalization* of q (or q is said to be a *proper instance* of p) and denoted by $p \succ q$. If both $p \succeq q$ and $q \succeq p$ hold, then p and q are said to be *variants* and denoted by $p \equiv q$. In what follows, we do not distinguish any tree patterns which are variants. The following property is called the *compactness* of tree pattern languages [2].

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 $i \in \{1, \ldots, k\}$.*

For a nonempty set S of tree patterns, a tree pattern p is said to be a *generalization* of S if $p \succeq q$ for any q in S. A *least generalization* of S, denoted by $lg(S)$, is a generalization p of S such that $p \preceq q$ for every generalization q of S. Here, we introduce a special tree pattern, *null tree pattern*, denoted by $\perp$. As a convention, we define that $\perp \prec p$ for any non-null tree pattern p, $\perp \equiv \perp$, $size(\perp) = \infty$, $lg(\emptyset) = \perp$, and $L(\perp) = \emptyset$. For any set S of tree patterns, $lg(S)$ always exists and is unique modulo $\equiv$. Furthermore, the following properties hold (see e.g. [10, 4]).

Proposition 2. *Let p, q be tree patterns and S_1, S_2 be finite sets of tree patterns. Then the following propositions hold.*

1. *If $p \succeq q$ then $size(q) \geq size(p)$.*
2. *If $p \succeq lg(S_1)$ then $L(p) \supseteq S_1$.*
3. *If $S_1 \supseteq S_2$ then $lg(S_1) \succeq lg(S_2)$.*
4. *If $p \succeq q$ then $L(p) \supseteq L(q)$.*
5. *If $|\Sigma| \geq 2$ then $lg(L(p)) \equiv p$ for any tree pattern p.*

3 Consistency problem

First, we show that the consistency problem for the hypothesis space $\mathcal{TP}^k$ ($k \geq 2$) is NP-complete. To show the NP-completeness, we use a transformation from the consistency problem for k-term DNF to that for $\mathcal{TP}^k$.

Let T and F be mutually disjoint nonempty finite sets of ground tree patterns. A set H of tree patterns is *consistent with* $\langle T, F\rangle$ if $T \subseteq L(H)$ and $F \cap L(H) = \emptyset$. The *consistency problem for $\mathcal{TP}^k$* is a problem to decide whether there exists a hypothesis $H \in \mathcal{TP}^k$ consistent with given positive and negative examples $\langle T, F\rangle$.

Let $U = \{u_1, \ldots, u_n\}$ ($n \geq 1$) be a set of boolean variables. We call a variable u_j or its negation $\bar{u}_j$ a *literal*. A *k-term DNF* is a disjunction $C_1 + \cdots + C_l$ of at most k terms, where each term C_i is a conjunction of literals. We denote by a vector $b = (b_1, \ldots, b_n) \in \{0,1\}^n$ the assignment that assigns b_j to u_j for every $1 \leq j \leq n$. A k-term DNF is *consistent with* a pair $\langle T, F\rangle$ of sets of assignments if it is true under all the assignments in T and false under all the assignments in F. The *consistency problem for k-term DNF* is a problem to decide whether there exists a k-term DNF H consistent with given a pair $\langle T, F\rangle$ of mutually disjoint sets of assignments.

In what follows, we use a special form of tree patterns. Let f, g be function symbols of arity n and a be a constant. For any $1 \leq i \leq n$, we define tree patterns $\mathbf{1}_1, \ldots, \mathbf{1}_n$ recursively as $\mathbf{1}_i = f(a, \ldots, a, \mathbf{1}_{i-1})$, where $\mathbf{1}_0 = a$. Similarly, we define $\mathbf{0}_1, \ldots, \mathbf{0}_n$ as $\mathbf{0}_i = g(a, \ldots, a, \mathbf{1}_{i-1})$. Note that $\mathbf{1}_i$ and $\mathbf{0}_i$ are the same tree patterns except their principal functors. We associate an assignment $b = (b_1, \ldots, b_n)$ with a ground tree pattern $\mathbf{b} = f(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, where $\mathbf{b}_i$ is either $\mathbf{0}_i$ or $\mathbf{1}_i$ according to $b_i \in \{0, 1\}$. We will use this informal notation without notice.

For tree patterns p and q, an *occurrence* of p in q is represented by a string α of positive integers defined as follows: if $p = q$ then p occurs at the empty string ε in p; if $q = f(q_1, \ldots, q_m)$ and p occurs at α in q_i $(1 \leq i \leq m)$ then p occurs at $i\alpha$ in q. A tree pattern p is said to be *regular* if any variable occurs at most once in p.

Lemma 3. *Let T and F be mutually disjoint sets of assignments, and $T' = \{\mathbf{b} \mid b \in T\}$ and $F' = \{\mathbf{b} \mid b \in F\}$ be the sets of ground tree patterns. Then, there is a k-term DNF consistent with $\langle T, F\rangle$ if and only if there is a set of at most k regular tree patterns consistent with $\langle T', F'\rangle$.*

Lemma 4. *Let S be any set of ground tree patterns of the form $f(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for some $(b_1, \ldots, b_n) \in \{0, 1\}^n$. Then, for any tree pattern p there is a regular tree pattern $\hat{p}$ such that $L(p) \cap S = L(\hat{p}) \cap S$.*

Theorem 5. *The consistency problem for $\mathcal{TP}^k$ is* NP*-complete for any $k \geq 2$.*

Proof. By Proposition 2, if a tree pattern language $L(p)$ has a nonempty intersection with a set $T \subseteq \mathcal{TP}(\Sigma)$ then $size(p) \leq size(t)$ for some $t \in T$. Thus, it is easy to see that the consistency problem for $\mathcal{TP}^k$ is in NP because there exists a nondeterministic algorithm that simply guesses at most k tree patterns $p_1, \ldots, p_l$ of size at most $\max\{size(t) \mid t \in T\}$ and checks in polynomial time whether $T \subseteq L(H)$ and $F \cap L(H) = \emptyset$, where $H = \{p_1, \ldots, p_l\}$.

We transform the consistency problem for k-term DNF to the consistency problem for $\mathcal{TP}^k$. Let T and F be mutually disjoint sets of assignments over $U = \{u_1, \ldots, u_n\}$ in an arbitrary instance of the consistency problem for k-term DNF. We associate T and F with sets of ground tree patterns $T' = \{\mathbf{b} \mid b \in T\}$ and $F' = \{\mathbf{b} \mid b \in F\}$.

By restricting the hypothesis space to the class of sets of at most k regular tree patterns, we obtain Lemma 3. From Lemma 4, we can remove the restriction that all the tree patterns in a consistent hypothesis should be regular. Hence, it immediately follows from Lemma 3 that there exists a k-term DNF consistent with $\langle T, F\rangle$ if and only if there exists a set $H' \in \mathcal{TP}^k$ consistent with $\langle T', F'\rangle$.

It is not difficult to see that the transformation can be computed in deterministic log space. By Pitt and Valiant [8], the consistency problem for k-term DNF is NP-complete for any $k \geq 2$. This completes the proof. $\square$

Proof of Lemma 1. Suppose that there exists a k-term DNF $H = C_1 + \cdots + C_l$ consistent with $\langle T, F\rangle$ and C_i $(1 \leq i \leq l)$ does not contain both a variable and its negation. We may assume $T \neq \emptyset$ and $F \neq \emptyset$ without loss of generality. We associate each term C_i $(1 \leq i \leq l)$ with a regular tree pattern $p_i = f(q_1, \ldots, q_n)$, where for every $1 \leq j \leq n$,

$$q_j = \begin{cases} \mathbf{1}_j, & \text{if } C_i \text{ contains } u_j, \\ \mathbf{0}_j, & \text{if } C_i \text{ contains } \bar{u}_j, \\ X_j, & \text{otherwise,} \end{cases}$$

and $X_1, \ldots, X_n$ are mutually distinct variables. We then set $H' = \{p_1, \ldots, p_l\}$. For any assignment $b = (b_1, \ldots, b_n) \in T \cup F$ and the corresponding ground tree pattern $\mathbf{b} = f(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, we can easily see that C_i is true under b if and only if $\mathbf{b} \in L(p_i)$. Thus, if H is consistent with $\langle T, F\rangle$ then $H' = \{p_1, \ldots, p_l\}$ is consistent with $\langle T', F'\rangle$.

We show the converse. Let $H' = \{p_1, \ldots, p_l\}$ be a set of at most k regular tree patterns reduced with respect to T' and consistent with $\langle T', F'\rangle$. We construct a k-term DNF H as follows: For each $1 \leq i \leq l$, p_i is a generalization of some member in T' because H' is reduced with respect to T'. Since $F' \neq \emptyset$, p_i is not a variable. Moreover, it follows from $T' \neq \emptyset$ that p_i must be a tree pattern of the form $f(q_1, \ldots, q_n)$ satisfying either $\mathbf{0}_j \prec q_j$ or $\mathbf{1}_j \prec q_j$ for every $1 \leq j \leq n$. Then, we associate p_i with a term $C_i = L_1 \cdots L_n$ such that for every $1 \leq j \leq n$,

$$L_j = \begin{cases} u_j, & \text{if } \mathbf{0}_j \not\preceq q_j \text{ and } \mathbf{1}_j \preceq q_j, \\ \bar{u}_j, & \text{if } \mathbf{0}_j \preceq q_j \text{ and } \mathbf{1}_j \not\preceq q_j, \\ 1, & \text{if } \mathbf{0}_j \preceq q_j \text{ and } \mathbf{1}_j \preceq q_j. \end{cases}$$

By construction, if C_i is true under $b = (b_1, \ldots, b_n)$ then $\mathbf{b}_1 = q_1\theta_1, \ldots, \mathbf{b}_n = q_n\theta_n$ for some substitutions $\theta_1, \ldots, \theta_n$. Since p_i is regular, the domains of $\theta_1, \ldots, \theta_n$ are mutually disjoint. We can take a substitution $\theta = \theta_1 \cup \cdots \cup \theta_n$ so that $\mathbf{b} = p_i\theta$ holds. Thus, $\mathbf{b} \in L(p_i)$. It is easy to verify that C_i is true under $b \in \{0,1\}^n$ if and only if $\mathbf{b} \in L(p_i)$. We then set $H = C_1 + \cdots + C_l$. It is clear that if H' is consistent with $\langle T', F'\rangle$ then H is also consistent with $\langle T, F\rangle$. Consequently, the claim is proved. $\square$

Proof of Lemma 2. Assume $L(p) \cap S \neq \emptyset$ without loss of generality. Let $p = f(p_1, \ldots, p_n)$ and $\mathbf{b} = f(\mathbf{b}_1, \ldots, \mathbf{b}_n) \in L(p) \cap S$. Suppose that some variable X

occurs at least twice in p, say, X occurs at α in p_i and at β in p_j, where i and j are possibly the same. If α and β are both ε, then X must match both $\mathbf{b}_i$ and $\mathbf{b}_j$. However, it is impossible since $\mathbf{b}_i \neq \mathbf{b}_j$ if $i \neq j$. Therefore, at least one of α and β, say α, is not ε, that is, α is a properly internal occurrence in both $\mathbf{b}_i$ and p_i. Let s be the subexpression of $\mathbf{b}$ occurring at α in $\mathbf{b}_i$. Let $\mathbf{b}' = f(\mathbf{b}'_1, \ldots, \mathbf{b}'_n)$ be an arbitrary member in $L(p) \cap S$. Then, the subexpression of $\mathbf{b}'_i$ occurring at α is also s because $\mathbf{b}_i$ and $\mathbf{b}'_i$ are either $\mathbf{0}_i$ or $\mathbf{1}_i$, and exactly the same at any internal position. For this reason, we can obtain a tree pattern $p\sigma$ that does not contain the variable X by applying the substitution $\sigma = \{X/s\}$ to p. Since $\mathbf{b}' \in L(p\sigma)$ whenever $\mathbf{b}' \in L(p)$, we have $L(p\sigma) \cap S = L(p) \cap S$. In the same way, we can obtain a regular tree pattern $\hat{p}$ with $L(p) \cap S = L(\hat{p}) \cap S$ from the given tree pattern p by removing all the variables occurring more than once. $\square$

Theorem 5 states that there is no algorithm for solving the consistency problem for $\mathcal{TP}^k$ ($k \geq 2$) in polynomial time unless P = NP. Note that the problem remains NP-complete even in the case where the class $\mathcal{TP}^k$ has the compactness, that is, $|\Sigma| \geq k + 1$.

4 A polynomial time algorithm using queries

In this section, we give an algorithm for solving the consistency problem for $\mathcal{TP}^2$ using queries. We also show that the algorithm runs in time polynomial in the size of given examples. The algorithm starts with the least generalization of given positive examples T then gradually refines it to approximate a fellow, a tree pattern p, of a hypothesis in $\mathcal{TP}^2$. The other fellow of the hypothesis is approximated as $lg(T - L(p))$. The search process is one-way, that is, without backtracking. This one-way search is accomplished by additional information obtained from making queries.

4.1 Queries and minimally consistent oracles

The types of queries we use in this paper are subset queries and membership queries. A *subset query* is to propose a tree pattern p and query whether $L(p)$ is a subset of a target language. A *membership query* is to propose a ground tree pattern α and query whether α is in a target language. A minimally consistent oracle is a device which answers each query as defined bellow.

Let T and F be mutually disjoint nonempty finite sets of ground tree patterns. A hypothesis space $\mathcal{TP}^k$ ($k \geq 1$) is said to be *least* with respect to $\langle T, F\rangle$ if there exists a hypothesis in $\mathcal{TP}^k$ which is consistent with $\langle T, F\rangle$ but no hypothesis in $\mathcal{TP}^{k-1}$ is consistent with $\langle T, F\rangle$. Note that, for any disjoint nonempty finite sets T and F, T itself is in $\mathcal{TP}^{|T|}$ and consistent with $\langle T, F\rangle$. Thus there always exists the least hypothesis space with respect to $\langle T, F\rangle$, since $k < k'$ implies $\mathcal{TP}^k \subset \mathcal{TP}^{k'}$.

A *subset oracle* is a device which receives a tree pattern as its input then returns "Yes" or "No" as its output according to some mapping from tree patterns to {"Yes", "No"}. A subset oracle is said to be *consistent* with a hypothesis H in $\mathcal{TP}^k$ if it returns "Yes" whenever it receives a tree pattern p such that $L(p) \subseteq L(H)$ and "No" otherwise. Let T and F be mutually disjoint nonempty finite sets of ground tree patterns. A subset oracle is said to be *minimally consistent* with $\langle T, F\rangle$ if it is consistent with some hypothesis H in the least hypothesis space with respect to $\langle T, F\rangle$ such that H is consistent with $\langle T, F\rangle$. Note that the subset oracle is not required to return a *counter example* [1]. That is , the subset query we consider in this paper corresponds to a *restricted subset query* in Angluin's framework.

A minimally consistent membership oracle is also defined similarly. A *membership oracle* is a device which receives a ground tree pattern as its input then returns "Yes" or "No" as its output according to some mapping from ground tree patterns to {"Yes", "No"}. A membership oracle is said to be *consistent* with a hypothesis H in $\mathcal{TP}^k$ if it returns "Yes" for any element in $L(H)$ and "No" for any element in $\mathcal{TP}(\Sigma) - L(H)$. Let T and F be mutually disjoint nonempty finite sets of ground tree patterns. A membership oracle is said to be *minimally consistent* with $\langle T, F\rangle$ if it is consistent with some hypothesis H in the least hypothesis space with respect to $\langle T, F\rangle$ such that H is consistent with $\langle T, F\rangle$.

4.2 Refinements of a tree pattern

For a tree pattern p, an instance $p\theta$ is said to be a *refinement* of p if one of the following holds:

1. $\theta = \{X/Y\}$, where X and Y are distinct variables occurring in p.
2. $\theta = \{X/f(X_1, X_2, \ldots, X_n)\}$ for some function symbol f with arity n in Σ, $n \geq 0$, where X is a variable that occurs in p and $X_1, \ldots, X_n$ are mutually distinct variables not occurring in p.

A *refinement operator* ρ over $\mathcal{TP}$ is a mapping from tree patterns to their refinements, that is, for a tree pattern p,

$$\rho(p) = \{p\theta \mid \theta \text{ satisfies one of the above two conditions }\}.$$

The refinement operator defined above is *complete* in the sense of [3], that is, the following proposition holds.

Proposition 6. *For any tree patterns p and q, if $p \succ q$ then there exists a finite sequence $r_0, r_1, \ldots, r_m$ $(m \geq 1)$ of tree patterns such that $p = r_0$, $q = r_m$, and $r_i \in \rho(r_{i-1})$ for any $1 \leq i \leq m$.*

In other words, for any tree pattern p, any proper instance of p can be obtained by finitely many applications of ρ to p. Furthermore, from the definition of ρ, it is clear that the following propositions hold.

Proposition 7. *For any tree pattern p and its refinement $q \in \rho(p)$, $size(q) \geq size(p) + 1$.*

Proposition 8. *Let n be the number of different variables appearing in a tree pattern p. Then it holds that $|\rho(p)| \leq n|\Sigma| + n(n-1)/2$.*

4.3 The algorithm and its correctness and complexity

The discussion developed in this section fully uses the compactness of tree pattern languages and Proposition 2 and our target is to search $\mathcal{TP}^2$ for a consistent hypothesis. Thus, in what follows, we assume that $|\Sigma| \geq 3$.

Figure 1 is the main procedure of the algorithm. The sub-procedure RS appears in Figure 2. In what follows, we assume that T and F are mutually disjoint nonempty finite sets of ground tree patterns.

Procedure: $FCH(T, F)$
Input: T, F: Mutually disjoint nonempty finite sets of ground tree patterns.
Given: A subset oracle which is minimally consistent with $\langle T, F \rangle$.
Output: A consistent hypothesis in $\mathcal{TP}^2$ or "Fail"
begin
  if $\{lg(T)\}$ is consistent with $\langle T, F \rangle$ **then**
    return $\{lg(T)\}$;
  else
    return $RS(T, F, lg(T))$;
end

Fig. 1. A procedure for finding a consistent hypothesis

If $\{lg(T)\}$ is consistent with $\langle T, F \rangle$, the algorithm outputs $\{lg(T)\}$ which is in $\mathcal{TP}^1 \subseteq \mathcal{TP}^2$. Thus, for the present, we focus on the case where a consistent hypothesis is in $\mathcal{TP}^2$ but not in $\mathcal{TP}^1$.

Let $\{p_1, p_2\}$ be a set of tree patterns which is reduced with respect to T. A tree pattern r is said to be *general enough* with respect to T and $\{p_1, p_2\}$ if and

Procedure: $RS(T, F, P)$
Input: T, F: Mutually disjoint nonempty finite sets of ground tree patterns.
P: A tree pattern. ($lg(T)$ at the initial call)
Given: A subset oracle which is minimally consistent with $\langle T, F\rangle$.
Output: A consistent hypothesis in $\mathcal{TP}^2$ or "Fail"
begin
if the pair $\{P, lg(T - L(P))\}$ is consistent with $\langle T, F\rangle$ **then**
return the pair $\{P, lg(T - L(P))\}$;
for each $R \in \rho(P)$ **do**
Make a subset query with $lg(T - L(R))$;
if the answer is "Yes" **then**
return $RS(T, F, R)$;
return "Fail";
end

Fig. 2. A recursive search procedure for a consistent hypothesis in $\mathcal{TP}^2 - \mathcal{TP}^1$

only if $r \succeq lg(T - L(p_i))$ for some $i \in \{1, 2\}$. From the definition, $lg(T - L(p_i))$ itself is general enough. Furthermore, since $lg(T) \succeq lg(T - L(p_i))$ follows from Proposition 2, the next proposition holds.

Proposition 9. *Let $\{p_1, p_2\}$ be a set of tree patterns reduced with respect to T. Then, $lg(T)$ is general enough with respect to T and $\{p_1, p_2\}$.*

The role of subset queries made by the algorithm is characterized by the next lemma and corollary. That is, the subset queries are used for testing whether the current refinement is general enough.

Lemma 10. *Let $\{p_1, p_2\}$ be a set of tree patterns reduced with respect to T. Then, a tree pattern r is general enough with respect to T and $\{p_1, p_2\}$ if and only if $lg(T - L(r))) \preceq p_i$ for some $i \in \{1, 2\}$.*

Proof. Suppose that r is general enough with respect to T and $\{p_1, p_2\}$, that is, $r \succeq lg(T - L(p_i))$ for some $i \in \{1, 2\}$. Since $L(r) \supseteq T - L(p_i)$ follows from Proposition 2, it holds that $L(p_i) \supseteq T - L(r)$. Thus, it follows from Proposition 2 that $p_i \equiv lg(L(p_i)) \succeq lg(T - L(r))$ for some $i \in \{1, 2\}$.

Conversely, suppose that $lg(T - L(r)) \preceq p_i$ for some $i \in \{1, 2\}$. Since $T - L(r) \subseteq L(p_i)$ follows from Proposition 2, it holds that $T - L(p_i) \subseteq L(r)$. Thus, it follows from Proposition 2 that $lg(T - L(p_i)) \preceq lg(L(r)) \equiv r$ for some $i \in \{1, 2\}$. $\square$

From Proposition 1, the next corollary follows.

Corollary 11. *Let $\{p_1, p_2\}$ be a set of tree patterns reduced with respect to T. Then, a tree pattern r is general enough with respect to T and $\{p_1, p_2\}$ if and only if $L(lg(T - L(r))) \subseteq L(p_1) \cup L(p_2)$.*

The next lemma ensures the termination of the algorithm with a correct output when the least hypothesis space with respect to $\langle T, F\rangle$ is $\mathcal{TP}^2$.

Lemma 12. *Let $\{p_1, p_2\}$ be a set of tree patterns reduced with respect to T and consistent with $\langle T, F\rangle$. If $r \equiv lg(T - L(p_i))$ for some $i \in \{1, 2\}$, then the pair $\{r, lg(T - L(r))\}$ of tree patterns is consistent with $\langle T, F\rangle$.*

Proof. Without loss of generality, we may assume that $i = 1$. Furthermore, since it is clear that $T \subseteq L(r) \cup L(lg(T - L(r)))$, it is sufficient to show that

$$F \cap (L(r) \cup L(lg(T - L(r)))) = \emptyset.$$

Since $\{p_1, p_2\}$ is reduced with respect to T, it holds that $T - L(p_1) \subseteq L(p_2)$. Hence, it follows from Proposition 2 that $r \equiv lg(T - L(p_1)) \preceq lg(L(p_2)) \equiv p_2$. From Proposition 2, it holds that $L(r) \subseteq L(p_2)$.

On the other hand, applying Proposition 2 to the reflexive relation $lg(T - L(p_1)) \succeq lg(T - L(p_1))$, we get $L(lg(T - L(p_1))) \supseteq T - L(p_1)$. This implies that $L(p_1) \supseteq T - L(lg(T - L(p_1))) = T - L(r)$. Hence, it follows from Proposition 2 that $p_1 \equiv lg(L(p_1)) \succeq lg(T - L(r))$. Thus, from Proposition 2, it holds that $L(p_1) \supseteq L(lg(T - L(r)))$.

As a consequence, it holds that $L(r) \cup L(lg(T - L(r))) \subseteq L(p_1) \cup L(p_2)$. This completes the proof of the lemma, since it holds that $F \cap (L(p_1) \cup L(p_2)) = \emptyset$ by the assumption ($\{p_1, p_2\}$ is consistent with $\langle T, F\rangle$). $\square$

Theorem 13. *Let T and F be mutually disjoint nonempty finite sets of ground tree patterns. For any minimally consistent subset oracle with $\langle T, F\rangle$, the procedure FCH returns a hypothesis in $\mathcal{TP}^2$ which is consistent with $\langle T, F\rangle$ if such a hypothesis exists in $\mathcal{TP}^2$ and returns "Fail" otherwise.*

Proof. First, we consider the case where the least hypothesis space with respect to $\langle T, F\rangle$ is $\mathcal{TP}^1$. If there exists a consistent hypothesis $\{p\} \in \mathcal{TP}^1$, then both $L(p) \supseteq T$ and $L(p) \cap F = \emptyset$ hold. From Proposition 2, it holds that $p \succeq lg(T)$. Since $L(p) \supseteq L(lg(T))$ follows from Proposition 2, it holds that $L(lg(T)) \cap F$ is also empty. From the definition of the least generalization, it holds that $L(lg(T)) \supseteq T$. Thus, $\{lg(T)\}$ is consistent with $\langle T, F\rangle$. Conversely, if $\{lg(T)\}$ is consistent with $\langle T, F\rangle$, then there exists the hypothesis $\{lg(T)\}$ in $\mathcal{TP}^1$. Hence, $\mathcal{TP}^1$ is the least hypothesis space with $\langle T, F\rangle$ if and only if $\{lg(T)\}$ is consistent with respect to $\langle T, F\rangle$. Thus, the procedure FCH works correctly if the least hypothesis space with respect to $\langle T, F\rangle$ is $\mathcal{TP}^1$.

Next, assume that the least hypothesis space with respect to $\langle T, F\rangle$ is $\mathcal{TP}^2$. Then, any subset oracle which is consistent with some $H \in \mathcal{TP}^2$ is given to the algorithm where $H = \{p_1, p_2\}$ is consistent with $\langle T, F\rangle$. If H is not reduced with respect to T, then either $\{p_1\}$ or $\{p_2\}$ is consistent with $\langle T, F\rangle$. Since both of them are in $\mathcal{TP}^1$, this contradicts the current assumption. Thus, we may assume that H is reduced with respect to T.

In this case, the procedure FCH calls the sub-procedure RS with inputs T, F, and $lg(T)$. From Proposition 9, the input $lg(T)$ is general enough with respect to T and $\{p_1, p_2\}$. Furthermore, every recursive call of RS, $RS(T, F, R)$, is made with the input R such that $L(lg(T - L(R))) \subseteq L(p_1) \cup L(p_2)$. Thus, from Corollary 11, R is also general enough with respect to T and $\{p_1, p_2\}$. Thus, every call $RS(T, F, P)$ of RS is made with the input P that is general enough with respect to T and $\{p_1, p_2\}$. Hence, it holds that $P \succeq lg(T - L(p_i))$ for some $i \in \{1, 2\}$. From Lemma 12, if $P \equiv lg(T - L(p_i))$ then $\{P, lg(T - L(P))\}$ is consistent with $\langle T, F\rangle$. Thus, the hypothesis $\{P, lg(T - L(P))\}$ in $\mathcal{TP}^2$ is outputted. On the other hand, from Proposition 6, if $P \succ lg(T - L(p_i))$ then there exists a finite sequence $r_0, r_1, \ldots, r_m$ such that $r_0 \equiv P$, $r_m \equiv lg(T - L(p_i))$, and $r_i \in \rho(r_{i-1})$ for any $1 \geq i \geq m$. Thus, even in the worst case, P is refined up to $lg(T - L(p_i))$ for some $i \in \{1, 2\}$. This ensures that the both procedure RS and FCH terminate with the output, $\{lg(T - L(p_i)), lg(T - L(lg(T - L(p_i))))\}$, which is consistent with $\langle T, F\rangle$ and in $\mathcal{TP}^2$ even in the worst case. This completes the proof for the case where the least hypothesis space with respect to $\langle T, F\rangle$ is $\mathcal{TP}^2$.

Finally, we consider the case where the least hypothesis space is $\mathcal{TP}^k$ for some $k \geq 3$. In this case, there is no hypothesis in $\mathcal{TP}^2$ which is consistent with $\langle T, F\rangle$. On the other hand, by the structure of the procedures, it is obvious that if FCH outputs a hypothesis in $\mathcal{TP}^2$ then the hypothesis is consistent with $\langle T, F\rangle$. Thus, if the procedure terminates finitely and makes some output, then the output must be "Fail". The termination of the procedure is ensured by the next theorem. □

Theorem 14. *The procedure FCH terminates in time polynomial in $size(T \cup F)$.*

Proof. From the discussion in the proof of the previous theorem, the least hypothesis space with respect to $\langle T, F\rangle$ is $\mathcal{TP}^1$ if and only if $lg(T)$ is consistent with $\langle T, F\rangle$. The least generalization $lg(T)$ can be calculated in time polynomial in $size(T)$ and also tests of the consistency of $lg(T)$ on $\langle T, F\rangle$ terminates in time polynomial in $size(T \cup F)$. Thus, if the least hypothesis space is $\mathcal{TP}^1$, then the statement of the theorem is correct.

Suppose that the least hypothesis space is $\mathcal{TP}^k$ for some $k \geq 2$ and H is a hypothesis in $\mathcal{TP}^k$ which is consistent with $\langle T, F\rangle$. Since H is in the least hypothesis space, H is reduced with respect to T. On the other hand, in this case, the procedure RS is called and recursively searches $\mathcal{TP}^2$ for a hypothesis. For each recursive call $RS(T, F, R)$ made in the procedure call $RS(T, F, P)$, since R is a refinement of P, it follows from Proposition 7 that $size(R) > size(P)$. Thus, at some stage of the recursive search, the size of the input tree pattern exceeds the maximum size, say ℓ, of elements in T. Let P be such an input tree pattern. Then, for any $R \in \rho(P)$ and $\alpha \in T$, it follows from Proposition 2 and Proposition 7 that $R \not\succeq \alpha$. Thus, it holds that $lg(T - L(R)) \equiv lg(T)$ for every $R \in \rho(P)$. If the answer from the oracle for the subset query with $lg(T - L(R))$ is "Yes", that is, $L(lg(T)) \subseteq L(H)$ holds, then it follows from Proposition 1 that $lg(T) \preceq p$ for some $p \in H$. This contradicts that H is reduced with respect to T, since it follows from Proposition 2 that $T \subseteq L(p)$. Thus, the answer from the oracle for the subset query with $lg(T - L(R))$ is "No" for every $R \in \rho(P)$. Hence, RS is called recursively at most $\ell - size(lg(T))$ times, since P is set to $lg(T)$ initially.

For any tree pattern p and a ground tree pattern α, it is clear that if the number of different variables appearing in p is larger than the size of α, then $p \not\succeq \alpha$. Thus, from the above discussion, the number of different variables appearing in any input tree pattern P does not exceeds ℓ. Thus, from Proposition 8, the number of possible refinements of P is bounded by a polynomial in ℓ. Furthermore, it is also clear that the calculation of $lg(T - L(P))$ terminates in time polynomial in $size(T)$ and the test if $\{P, lg(T - L(P))\}$ terminates in time polynomial in $size(T \cup F)$. This completes the proof of the theorem. $\square$

4.4 Replacing a subset query with membership queries

In this section, we show that the procedure which uses several membership queries instead of a subset query also works well under some condition for Σ.

Lemma 15. *Suppose that Σ ($|\Sigma| \geq 3$) contains at least one function with non-zero arity. Then, for any tree patterns r, p_1, p_2, there exists a set $G(r)$ of $n + 1$ ground instances of r such that $G(r) \subseteq L(p_1) \cup L(p_2)$ if and only if $r \preceq p_i$ for some $i \in \{1, 2\}$, where n is the number of different variables occurring in r.*

Proof. The *if* direction of the statement is trivial. Thus, we show the *only if* direction. Without loss of generality, we may assume that Σ contains one function with arity 1, say $f(\_)$, and two constants a, b. If r is ground then the proof is trivial. Thus, we assume that r contains n variables, say $x_1, \ldots, x_n$.

Consider the substitutions $\theta_0, \theta_1, \ldots, \theta_n$ which replace each variable by some ground tree pattern as defined in the following table and define $G(r)$ as the set $\{r\theta_0, r\theta_1, \ldots, r\theta_n\}$.

$$\begin{array}{c|cccc} & x_1 & x_2 & \cdots & x_n \\ \hline \theta_0 & f(a) & f(f(a)) & \cdots & \overbrace{f(f(\cdots f}^{n}(a)\cdots) \\ \theta_1 & a & b & \cdots & b \\ \theta_2 & b & a & \cdots & b \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_n & b & b & \cdots & a \end{array}$$

Note that, θ_0 replaces every variables with the mutually distinct ground tree patterns whose principal functors are different from a and b, θ_i ($i \neq 0$) replaces x_j by a if $i = j$, b otherwise. Such a substitution can be constructed even if the arity of f is greater than 1. Then, $lg(\{r\theta_0, r\theta_j\}) \equiv r$ for any $1 \leq j \leq n$. Thus, if $L(p_1)$ contains $r\theta_0$ and at least one $r\theta_j$ for some $1 \leq j \leq n$, then from Proposition 2 and Proposition 2 $p_1 \succeq r$. On the other hand, if $L(p_1)$ contains no $r\theta_j$ for any $1 \leq j \leq n$, then $L(p_2) \supseteq \{r\theta_1, \ldots, r\theta_n\}$. Since it is clear that $lg(\{r\theta_1, \ldots, r\theta_n\}) \equiv r$, we have $p_2 \succeq r$. The dual discussion holds when $L(p_2)$ contains $r\theta_0$ or no $r\theta_j$ for any $1 \leq j \leq n$. As a consequence, if $S \subseteq L(p_1) \cup L(p_2)$ then it holds that $r \preceq p_i$ for some $i \in \{1, 2\}$. □

By the above lemma, the following holds as another corollary of Lemma 10.

Corollary 16. *Let $\{p_1, p_2\}$ be a set of tree patterns reduced with respect to T. Suppose that Σ ($|\Sigma| \geq 3$) contains at least one function with non-zero arity. Then, a tree pattern r is general enough with respect to T and $\{p_1, p_2\}$ if and only if $G(lg(T - L(r))) \subseteq L(p_1) \cup L(p_2)$.*

As a consequence, we can replace each subset query made in the procedure *RS* with $n+1$ membership queries, where n is the number of different variables occurring in $lg(T - L(r))$ for some tree pattern r. Since n is bounded by the maximal size of elements in T, the following theorem follows from Theorem 13 and Theorem 14.

Theorem 17. *Let T and F be mutually disjoint nonempty finite sets of ground tree patterns. There exists an algorithm $\mathcal{A}$ that, for any given minimally consistent membership oracle with $\langle T, F\rangle$, satisfies the following conditions:*

1. *$\mathcal{A}$ terminates in time polynomial in $size(T \cup F)$.*
2. *$\mathcal{A}$ outputs a hypothesis in $\mathcal{TP}^2$ which is consistent with $\langle T, F\rangle$ if such a hypothesis exists, and outputs "Fail" otherwise.*

5 Concluding remarks

In this paper, we presented an algorithm for solving the consistency problem for $\mathcal{TP}^2$ using queries. There may be a lot of opinions for the use of queries to solve the consistency problem. In some application domains such as motif-discovering in Molecular Biology, it is reasonable to use queries, since making a (membership) query can be regarded as making an experiment. However, there is controversy as to whether the *minimal* consistency of the oracle is reasonable. In this paper, we assume the minimality in order to avoid giving an oracle consistent with a hypothesis in $\mathcal{TP}^k$ for $k \geq 3$, while there exists a hypothesis in $\mathcal{TP}^2$ which is consistent with given $\langle T, F \rangle$. An algorithm which works well, for any oracle which is simply consistent with the given sets, might be ideal. However, it seems impossible to find a consistent hypothesis in $\mathcal{TP}^2$ via an oracle which answers according to a trivial consistent hypothesis T such that $|T| \geq 3$. It is our future work to consider more natural settings for solving consistency problem using queries.

Another interesting problem related to this work is the inductive inference with *refutation* [6]. A consistency problem is closely related to the refutability of a hypothesis space. Our work might be understood as an approach to construct an efficient inductive inference algorithm with refutation. We would like to clear the relation between the consistency problem and space refutable inference.

References

1. D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
2. H. Arimura, T. Shinohara, and S. Otsuki. A polynomial time algorithm for finding finite unions of tree pattern languages. In *Proc. of the 2nd International Workshop on Nonmonotonic and Inductive Logics*, 1991. LNAI 659.
3. 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 P. Enjalbert, E. Mayr, and K. W. Wagner, editors, *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 649–660. Springer-Verlag, 1994. LNCS 775.
4. J-L.Lassez, M. J. Maher, and K. Marriott. Unification revisited. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pp. 587–625. Morgan Kaufmann, 1988.
5. S. Miyano, A. Shinohara, and T. Shinohara. Which classes of elementary formal systems are polynomial-time learnable? In S. Arikawa, A. Maruoka, and T. Sato, editors, *Proc. ALT '91*, pp. 139–150. JSAI, 1991.
6. Y. Mukouchi and S. Arikawa. Inductive inference machines that can refute hypothesis spaces. In K. P. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *Proc. ALT '93*, pp. 123–136. Springer-Verlag, 1993. LNAI 744.

7. B. K. Natarajan. *Machine Learning, A Theoretical Approach.* Morgan Kaufmann, 1991.
8. L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *JACM*, 35(4):965–984, 1988.
9. G. D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pp. 153–163. Edinburgh University Press, 1970.
10. J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pp. 135–152. Edinburgh University Press, 1970.
11. L. G. Valiant. A theory of the learnable. *Comm. ACM*, 27:1134–1142, 1984.