Proc. the 2nd International Workshop on Nonmonotonic and Inductive Inference (NIL'91),
LNAI 659, Springer-Verlag, pp. 118-131, Dec. 1991.

A Polynomial Time Algorithm for Finding Finite Unions of Tree Pattern Languages

Hiroki ARIMURA arim@ai.kyutech.ac.jp
Takeshi SHINOHARA shino@ai.kyutech.ac.jp
Setsuko OTSUKI otsuki@ai.kyutech.ac.jp

Department of Artificial Intelligence
Kyushu Institute of Technology
680-4, Kawazu, Iizuka 820, JAPAN

Abstract

A tree pattern is a structured pattern known as a term in formal logic, and a tree pattern language is the set of trees which are the ground instances of a tree pattern. In this paper, we deal with the class of tree languages whose language is defined as a union of at most k tree pattern languages, where k is an arbitrary fixed positive number. In particular, we present a polynomial time algorithm that, given a finite set of trees, to find a set of tree patterns that defines a minimal union of at most k tree pattern languages containing the given set. The algorithm can be considered as a natural extension of Plotkin's anti-unification algorithm, which finds a minimal single tree pattern language containing the given set. By using the algorithm, we can realize a consistent and conservative polynomial time inference machine that identifies the class of unions of k tree pattern languages in the limit from positive data for every $k > 0$.

1 Introduction

Inductive inference is a process to guess an unknown rule from its examples. In this paper, a rule we consider is a union of at most k tree pattern languages for a fixed positive number k.

A *tree pattern* is a structured pattern known as a *term* in formal logic, which is a fundamental data structure in logic programming and term rewriting systems. In this paper, we pay attention to the class of languages, called *tree pattern languages*, defined as the set of all the ground instances of a tree pattern, and the class of unions of tree pattern languages. Since the class of tree pattern languages has finite thickness, that is, for any constant tree there are only finitely many tree patterns containing it, an inference machine that guesses a minimal language explaining the examples correctly infers the unknown tree pattern from its examples [Ang80a, LMM88]. For instance, we assume that a rule is represented by a tree pattern. Suppose that the following trees are given as its examples:

$$app([],[],[]),\ app([b],[a],[b,a]),\ app([a],[],[a]),$$
$$app([],[a],[a]),\ app([a,b],[c,d],[a,b,c,d]).$$

Then, a single atom

$$app(x,y,z)$$

defines a minimal tree pattern language containing all the examples. it is known that this minimal tree pattern, called the *least general generalization* in Plotkin [Plo70], can be computed by using the *anti-unification (least generalization) algorithm* [LMM88, Plo70] in polynomial time.

In this paper, we will consider inductive inference in the case where examples are taken from k tree pattern languages, and pay attention to the problem of finding a set of at most k tree patterns that represents a minimal union containing given examples. For instance, a pair

$$\{app([], x, x),\ app([x|y], z, [x|w])\}$$

of two patterns represents a minimal union of two tree pattern languages containing the above examples.

On the other hand, for unions of string pattern languages, it is still not known whether minimal unions of string pattern languages can be computed in polynomial time [Shi83, Wri89b]. Only for unions of one-variable pattern languages, it is shown to be computable in polynomial time [Wri89a].

In this paper we study polynomial time inferability of unions of at most k tree pattern languages from positive data for every positive number k. After preparing basic notions for tree patterns and inductive inference in Section 2, we introduce the class of unions of at most k tree pattern languages in Section 3. We first prove that the class has the compactness with respect to containment, which plays an important role to prove the correctness of our inference algorithm, when the alphabet contains more than k symbols.

In Section 4, we present a polynomial time algorithm, called k-MMG. The fundamental idea of our inference algorithm comes from the work of Wright [Wri89b]; He proved that a minimal union containing the given set can be computed in polynomial time for unions of two one-variable pattern languages. However we can not apply his method directly to prove inferability of unions of tree patterns even for $k = 2$. Because the number of generalizations of the given finite sample may be exponential for tree patterns, while it is at most polynomial for one-variable patterns. In Section 5, we show that it is not necessary to check exponentially many all partitions of a sample, but it is sufficient to consider all pairs of constant trees selected from the sample. That is, a minimal union can be obtained from some pair of constant trees by using a polynomial time algorithm to compute maximal tree patterns consistent with respect to the sample. Hence, we prove that the class of unions of k tree pattern languages is polynomial time inferable from positive data.

2 Preliminaries

2.1 Tree pattern languages

A *ranked alphabet* is a pair $(A, arity)$ of an alphabet A and a mapping *arity* from A to nonnegative numbers. Hereafter, if it is clear from the context, we may say a ranked alphabet A instead of $(A, arity)$.

We define a set $\mathcal{T}_A$ of strings over A together with three symbols "(", ")", "," as the smallest set satisfying

1. $c \in \mathcal{T}_A$ for any zero-ary symbol $c \in A$, and

2. $f(t_1, \ldots, t_n) \in \mathcal{T}_A$ for any $t_1, \ldots, t_n \in \mathcal{T}_A$ and any n-ary symbol $f \in A$ ($n > 0$).

Let Σ be a fixed alphabet, whose elements $a, a_1, a_2, \ldots, f, g, \ldots$ are called *function symbols*, and X be a countable set disjoint from Σ, whose elements $x, x_1, x_2, \ldots$ are called *variables*. We assume that all of variables are zero-ary symbols. We denote by $\mathcal{TP}_\Sigma$ the set $\mathcal{T}_{\Sigma \cup X}$. Elements of $\mathcal{T}_\Sigma$ are called *trees over* Σ and elements of $\mathcal{TP}_\Sigma$ are called *tree patterns over* Σ.

A *node* is a finite string of positive integers. We denote by $\alpha\beta$ the concatenation of strings α and β. A node α is called an *ancestor* of a node β if there is some γ such that $\alpha\gamma = \beta$. Let p be a tree pattern $f(p_1, \ldots, p_n)$ over Σ ($n \geq 0$). Then, the label $p(\alpha)$ of a node α in p is defined as

$$p(\alpha) = \begin{cases} f & \text{, if } \alpha = \epsilon, \\ p_i(\beta) & \text{, if } \alpha = i\beta \text{ and } 1 \leq i \leq arity(f), \\ \text{undefined} & \text{, otherwise.} \end{cases}$$

The *subtree* p/α of p at a node α is defined as (1) $p/\alpha = p$ if $\alpha = \epsilon$, (2) $p/\alpha = p_i/\beta$ if $\alpha = i\beta$ and $1 \leq i \leq arity(f)$, (3) p/α is undefined otherwise. The *tree domain* of p is the set $\mathcal{D}_p = \{\alpha \mid t(\alpha) \text{ is defined for a node } \alpha\}$. For instance, for a tree pattern $p = f(x, f(a, b))$ and the domain $\mathcal{D}_p = \{\epsilon, 1, 2, 21, 22\}$, the label of p at the node 2 is $p(2) = f$ and the subtree of p at 2 is $p/2 = f(a, b)$. We can identify a tree pattern p to the label mapping $p : \mathcal{D}_p \to \Sigma \cup X$.

A *substitution* is a mapping θ from $\mathcal{TP}_\Sigma$ to themselves satisfying $\theta(f(p_1, \ldots, p_n)) = f(\theta(p_1), \ldots, \theta(p_n))$ for any $f(p_1, \ldots, p_n) \in \mathcal{TP}_\Sigma$ ($n \geq 0$). Any mapping from X to $\mathcal{TP}_\Sigma$ can be uniquely extended to a substitution over $\mathcal{TP}_\Sigma$.

We use equality symbol $=$ as syntactic identity. We define a relation $\leq'$ over tree patterns as $s \leq' t$ if $s = t\theta$ for some substitution θ. Then, we say a tree pattern s is an *instance* of a tree pattern t. We define $s <' t$ if $s \leq' t$ but $t \not\leq' s$, and $s \equiv' t$ if $s \leq' t$ and $t \leq' s$. Note that $s \equiv' t$ iff $s = t\theta$ for some renaming θ of variables. The following lemma is basic for the relation $\leq'$.

Lemma 1. $p \leq' q$ iff both of (1) and (2) hold.

1. For any $\alpha \in \mathcal{D}_q$, either $q(\alpha) \in X$ or $p(\alpha) = q(\alpha) \in \Sigma$.
2. For any $\alpha, \beta \in \mathcal{D}_q$, if $q(\alpha) = q(\beta) \in X$ then $p/\alpha = p/\beta$.

The relation $\leq'$ can be decided in polynomial time [Kan88]. The *size* $|p|$ of a tree pattern p is the total number of occurrences of variable symbols and function symbols in p. We call a finite set $S \subseteq \mathcal{T}_\Sigma$ a *sample*. The *size* $||S||$ of S is the total size of trees in S.

For a tree pattern $p \in \mathcal{TP}_\Sigma$, we defined the set, denoted by $L(p)$, of all ground instances of p as

$$L(p) = \{w \in \mathcal{T}_\Sigma \mid w \leq' p\}.$$

A set $L \subseteq \mathcal{T}_\Sigma$ is called a *tree pattern language* if $L = L(p)$ for some tree pattern p. Let p, q be tree patterns. We call $L(p)$ the *language defined by a tree pattern* p. We say p is equivalent to q if $p \equiv' q$.

Lemma 2. (Corollary 12 in [LMM88]) Assume that Σ contains more than one symbols. Then, $L(p) \subseteq L(q)$ iff $p \leq' q$.

The next lemma is important in Section 5. For a set $S \subseteq \mathcal{TP}_\Sigma$, a tree pattern p is the *greatest common instance* of S, denoted by $gci(S)$, if (1) p is a *common instance* of S, that is, $p \leq' q$ for any $q \in S$, and (2) p' is not a common instance of S for any $p <' p'$. There is only one $gci(S)$ up to renaming of variables, and the $gci(S)$ can be computed by the *unification algorithm* in linear time [Kan88, LMM88].

Lemma 3. The class $\mathcal{TPL}_\Sigma$ is closed under intersection.

Proof. The lemma immediately follows from that the $gci(S)$ is unique for set $S = \{p_1, \ldots, p_n\} \subseteq \mathcal{TP}_\Sigma$ $(n > 0)$. □

A tree pattern p is a *generalization* of a tree pattern q if $q \leq' p$. For a set S of tree patterns, p is the *least generalization* (least common anti-instance) of S, denoted by $lca(S)$, if (1) p covers S, that is, $q \leq' p$ for any $q \in S$, and (2) p' does not cover S for any $p' <' p$. There is only one $lca(S)$ up to renaming of variables.

Lemma 4. (Lemma 6 in [LMM88]) For any $w \in \mathcal{T}_\Sigma$, there are finitely many generalizations of w.

Lemma 5. (Lemma 8 in [LMM88]) Assume that Σ contains more than one symbols. Then, for any $p \in \mathcal{TP}_\Sigma$, $lca(L(p)) \equiv' p$.

Lemma 6. Assume that Σ contains more than one symbols and $S, T \subseteq \mathcal{TP}_\Sigma$. If $S \subseteq T$, then $lca(S) \leq' lca(T)$.

Lemma 7. (Huet's Anti-unification algorithm; Theorem 8 in [LMM88]) Let ϕ be any one-to-one mapping from $\mathcal{TP}_\Sigma \times \mathcal{TP}_\Sigma$ to X. We extends ϕ to the mapping λ from $\mathcal{TP}_\Sigma \times \mathcal{TP}_\Sigma$ to $\mathcal{TP}_\Sigma$ defined as, for any $s, t \in \mathcal{TP}_\Sigma$,

$$\lambda(s,t) = \begin{cases} f(\lambda(s_1,t_1),\ldots,\lambda(s_n,t_1)) & \text{, if } s = f(s_1,\ldots,s_n) \text{ and } t = f(t_1,\ldots,t_n) \text{ for } f \in \Sigma\ (n \geq 0), \\ \phi(s,t) & \text{, otherwise.} \end{cases}$$

Then, $\lambda(s,t)$ is $lca(\{s,t\})$ for any $s, t \in \mathcal{TP}_\Sigma$. Moreover, for a finite set $S = \{t_0, \ldots, t_k\} \subseteq \mathcal{TP}_\Sigma$ $(k \geq 0)$, the tree pattern $\lambda(\lambda(\ldots\lambda(\lambda(t_0,t_1),t_2)\ldots),t_k)$ is $lca(S)$.

Since a one-to-one mapping ϕ can be implemented to run in polynomial time in $||S||$ for a finite S, we can easily see that $lca(S)$ can be computed in polynomial time [Plo70, LMM88]. The anti-unification algorithm was originally proposed by Plotkin in [Plo70], and independently by Reynolds in [Rey70]. In [Plo70], it was called the *least general generalization* algorithm.

2.2 Inductive inference from positive data

We fix a finite alphabet Σ. An *indexed family of recursive languages* is a class of nonempty languages $\mathbf{C} = \{L_1, L_2, L_3, \ldots\}$ such that there is an algorithm that, given a string $w \in \Sigma^*$ and an index i, decides whether $w \in L_i$. A *positive presentation* σ of L is an infinite sequence $w_1, w_2, \ldots$ of strings such that $\{w_n \mid n > 0\} = L$.

An *inference machine* is an effective procedure M that requests a string and produces a conjecture at a time. Given a positive presentation $\sigma = w_1, w_2, \ldots$, M generates an infinite sequence $g_1, g_2, \ldots$ of conjectures as follows: it starts with the empty sample $S_0 = \emptyset$. When M makes the n-th request ($n > 0$), a string w_n is added to the sample. Then, M reads the current sample $S_n = \{w_1, \ldots, w_n\}$ and adds a conjecture g_n to the end of the sequence of conjectures; any conjecture g_n ($n > 0$) must be an index of $\mathbf{C}$. We say that M *identifies* $L_i \in \mathbf{C}$ *in the limit from positive data* if for any positive presentation σ of L_i, there is some g such that for all sufficiently large n, the n-th conjecture g_n is identical to g and $L_g = L_i$. A class of languages $\mathbf{C}$ is said to be *identifiable in the limit from positive data* if there is an inference machine M such that for any $L_i \in \mathbf{C}$, M identifies L_i in the limit from positive data.

An inference machine M is said to be *consistent* if for any $n > 0$, it always produces a conjecture g_n such that $S_n \subseteq L_{g_n}$, and to be *conservative* if for any $n > 0$, it does not change the last conjecture g_{n-1} whenever $S_n \subseteq L_{g_{n-1}}$.

$\mathbf{C}$ is said to be *consistently and conservatively identifiable in the limit from positive data with polynomial time updating conjectures* if there is an inference machine M that consistently and conservatively identifies $\mathbf{C}$ in the limit from positive data and there is some polynomial $q(\cdot,\cdot)$ such that for any size $|g|$ of the representation of the unknown language $L_g \in \mathbf{C}$ the time used by M between receiving the n-th example w_n and outputting the n-th conjecture g_n is at most $q(|g|, |w_1| + \cdots + |w_n|)$, where $|w_j|$ is the length of w_j [Ang79]. For our inference algorithm, the update time is bounded by the polynomial only in $|w_1| + \cdots + |w_n|$. Since we do not consider other criteria for polynomial time inference in this paper, we simply call the criterion we use *polynomial time inference from positive data*.

In [Ang79], Angluin showed a sufficient condition of polynomial time inferability from positive data for classes with finite thickness. A class $\mathbf{C}$ has *finite thickness*, called Condition 3 in [Ang80b], if the set $\{L \in \mathbf{C} \mid w \in L\}$ is finite for any string $w \in \Sigma^*$. We extend the condition for classes with finite elasticity [Wri89a]. A class $\mathbf{C}$ has *infinite elasticity* if there exist two infinite sequences $w_0, w_1, \ldots$ of strings and $L_1, L_2, \ldots$ of languages in $\mathbf{C}$ such that $w_i \notin L_i$ and $\{w_1, \ldots, w_{i-1}\} \subseteq L_i$. A class $\mathbf{C}$ has *finite elasticity* if $\mathbf{C}$ does not have infinite elasticity [Wri89a]. Clearly from the definition, finite thickness implies finite elasticity.

Theorem 8. ([Wri89a]) For any positive number k, if a class $\mathbf{C}$ of languages has finite elasticity, then the class of unions of at most k languages in $\mathbf{C}$ has finite elasticity. □

For an indexed family $\mathbf{C}$ of recursive languages, the *membership problem* for $\mathbf{C}$ is the decision problem stated as

Given: a string $w \in \Sigma^*$ and the index i of a language in $\mathbf{C}$.
Problem: determine whether $w \in L_i$.

The *minimal language problem* for $\mathbf{C}$ is the search problem defined as

Given: a sample $S \subseteq \Sigma^*$.
Problem: find an index i of a minimal language containing S, that is, an index i satisfying (i) $S \subseteq L_i$ and (ii) for any j, if $L_j \subset L_i$ then $S \not\subseteq L_j$.

We denote by $minl(S)$ an arbitrary minimal language containing a given set S.

Lemma 9. If a class $\mathbf{C}$ has finite elasticity, and both of the membership problem for $\mathbf{C}$ and the minimal language problem for $\mathbf{C}$ are computable in polynomial time with respect to $||S||$, then

```
procedure M;
      input: an infinite sequence w_1, w_2, ...  of strings;
      output: an infinite sequence g_1, g_2, ...  of guesses;
begin
      set g_0 to be the null index, i.e., L_{g_0} = ∅, set S = ∅ and set i = 0;
      repeat
            read the next example w_i and add it to S;
            if w_i ∉ L_{g_{i-1}} then let g_i be minl(S) else let g_i be g_{i-1};
            output g_i and let i be i + 1;
      forever;  /* main loop */
end.
```

Figure 1: Angluin's inference machine M.

the procedure M shown in Figure 1 infers $\mathbf{C}$ from positive data in polynomial time, where $||S||$ is the total length of the strings in S.

Proof. The proof proceeds in a similar way as that in Angluin [Ang79]. Let σ be a positive presentation of a target language L_k in $\mathbf{C}$. Let $g_0, g_1, \ldots$ be a subsequence of distinct guesses produced by M over Σ and $w_0, w_1, \ldots$ be an input data that cause these changes of guesses. If M over Σ does not converge, then these two sequences are infinite. Furthermore, they show infinite elasticity of $\mathbf{C}$ because M is consistent and conservative. Therefore, we can conclude that M converges to some guess g_N at a finite stage $N > 0$. Thus, $L_k \subseteq L_{g_i}$ for any $i \geq N$ because M is consistent. On the other hand, $L_k \supseteq L_{g_i}$ holds for any $i \geq N$ since M always outputs an index of a minimal language containing S_i. Hence, outputs converge to a correct index g_N such that $L_k = L_{g_N}$. It is obvious from assumptions that M runs in polynomial time in $||S||$. $\square$

3 Unions of tree pattern languages

In this section, we introduce the class $(\mathcal{TPL}_\Sigma)^k$ of unions of at most k tree pattern languages for every positive number k, and prove some preliminary results for them.

Let $k > 0$ be any fixed number. The *union defined by a set* $\{p_1, \ldots, p_k\}$ *of at most* k *tree patterns*, denoted by $L(\{p_1, \ldots, p_k\})$, is the union $L(p_1) \cup \cdots \cup L(p_k)$. We refer to the class of unions of at most k tree pattern languages as $(\mathcal{TPL}_\Sigma)^k$ and to the class of sets of at most k tree patterns as $(\mathrm{TP}_\Sigma)^k$. We may omit the subscript Σ if it is clear from context.

Lemma 10. For every $k > 0$, the membership problem for $(\mathcal{TPL}_\Sigma)^k$ is polynomial time decidable.

Proof. Let $\{p_1, \ldots, p_k\} \in (\mathcal{TP}_\Sigma)^k$. Then, by Lemma 2, we can determine whether $w \in L(\{p_1, \ldots, p_k\})$ by using a polynomial time term matching algorithm to check whether $w \preceq' p_i$ for some $1 \leq i \leq k$. $\square$

Lemma 11. For every $k > 0$, the class $(\mathcal{TPL}_\Sigma)^k$ has finite elasticity.

Proof. For $w \in \mathcal{T}_\Sigma$ and $p \in \mathcal{TP}_\Sigma$, $w \in L(p)$ iff $w \preceq' p$ by Lemma 2. Thus, Lemma 4 shows finite thickness of the class $\mathcal{TPL}_\Sigma$. Since finite thickness implies finite elasticity [Wri89a], $\mathcal{TPL}_\Sigma$ has finite elasticity. Hence, the conclusion immediately follows from Theorem 8. $\square$

The class $(\mathcal{TPL}_\Sigma)^k$ is *compact with respect to containment* if for any $L \in \mathcal{TPL}_\Sigma$ and any union $L_1 \cup \cdots \cup L_k \in (\mathcal{TPL}_\Sigma)^k$,

$$L \subseteq L_1 \cup \ldots \cup L_k \quad \Rightarrow \quad L \subseteq L_i \text{ for some } 1 \le i \le k.$$

Lassez et. al.[LM86] showed that if Σ is an infinite alphabet, then the class $\mathcal{TPL}_\Sigma$ of any finite unions of tree pattern languages over Σ is compact with respect to containment. Now, we refine the result of Lassez et. al. for a finite Σ. For a set S, we denote by $\sharp S$ the number of elements of S.

Proposition 12. (Arimura et. al. [ASO92]) For every $k > 0$, if Σ contains more than k symbols, then the class $(\mathcal{TPL}_\Sigma)^k$ is compact with respect to containment.

Proof. We show a sketch of the proof. Detailed descriptions will be found in [ASO92]. Assume that $L(p) \subseteq L(q_1) \cup \ldots \cup L(q_k)$ (Eq. 1). Suppose that $s > k$ for $s = \sharp\Sigma$. First, let $n \ge 0$ be the number of distinct variables occur in p and $t_1, \ldots, t_s$ be s constant trees with mutually distinct functors as their root labels. We make the set A_Σ of instances of p by substituting only constant trees $t_1, \ldots, t_s$ to variables of p. Clearly $A_\Sigma \subseteq L(p)$. Then, the following claim holds.

Claim . For any subset $B \subseteq A_\Sigma$ and any tree pattern language $L(q)$ such that $B \subseteq L(q)$, if $\sharp B > s^{n-1}$, then $L(p) \subseteq L(q)$ holds.

On the other hand, $\sharp A_\Sigma$ is exactly s^n, and one of tree pattern languages $L(q_i)$ in the right hand side of Eq. 1 contains at least $s^n/k > s^{n-1}$ elements of A_Σ. Hence, the language $L(q_i)$ contains $L(p)$ by Claim. $\square$

The above proposition is frequently used in this paper. We can not remove the restriction $\sharp\Sigma > k$ from Proposition 12. Because we can construct a counter example in the case where $\sharp\Sigma$ is less than or equal to k for every $k > 1$. We give an example for $k = 2$. Let $\Sigma = \{a, f\}$, where $arity(a) = 0$ and $arity(f) = 2$, and $x, a, f(x_1, x_2)$ be tree patterns over Σ. Then, $L(x) \subseteq L(a) \cup L(f(x_1, x_2))$ over Σ, but $L(x) \not\subseteq L(a)$ and $L(x) \not\subseteq L(f(x_1, x_2))$.

4 Finding a minimal union

In this section, we present a polynomial time algorithm k-MMG that, given a finite set S of trees, computes a set of k tree patterns which defines a minimal language containing S. By using the algorithm, we show the polynomial time inferability of the class $(\mathcal{TPL}_\Sigma)^k$ for every $k > 0$ under some restriction for the alphabet Σ.

The problem here is the minimal language problem for the class $(\mathcal{TPL}_\Sigma)^k$ with the compactness with respect to containment. The notion of k-mmg is a natural extension of that of the least generalization studied by Plotkin[Plo70].

Definition 1. For a set $S \subseteq \mathcal{T}_\Sigma$, we call a minimal language containing S within $(\mathcal{TPL}_\Sigma)^k$ a *k-minimal multiple generalization* (k-mmg) for every $k > 0$.

Let $S \subseteq \mathcal{T}_\Sigma$ be a finite set. In the case where $k = 1$, a 1-mmg coincides to the least generalization $lca(S)$ of S. Thus, it can be computed in polynomial time by using the anti-unification algorithm as we saw in Section 2. In the case where $k = 2$, Arimura et. al.[ASO92] showed that a 2-mmg

```
    procedure k-MMG(S); (k > 1)
        input: a finite set S ⊆ T_Σ of trees;
        output: a k-mmg of S;
    begin
1       find a reduced set p̄ = {p_1, ..., p_k} of
            exactly k tree patterns with respect to S;
2       if found then begin
3           for each i = 1, ..., k do begin
4               let S_i = S − L(p̄ − {p_i});
5               replace p_i in p̄ by q_i ≡' lca(S_i);
6           end  /* tightening process */
7           return p̄;
8       end;
9       if not found then return (k − 1)-mmg(S);
    end.
```

```
procedure 1-MMG(S);
begin
    return lca(S);
end.
```

Figure 2: Algorithms to compute a k-mmg(S) for every $k > 0$

of S can be computed in polynomial time in $||S||$. We extends the method for the case where $k > 2$.

For a set $S \subseteq \mathcal{T}_\Sigma$, a set $\bar{p}$ of tree pattern languages is *reduced with respect to* S if (1) $S \subseteq L(\bar{p})$, and (2) $S \not\subseteq L(\bar{q})$ for any proper subset $\bar{q}$ of $\bar{p}$. A reduced set $\bar{p}$ with respect to S is a *normal form* with respect to S if there is no $p \in \bar{p}$ such that $lca(S - L(\bar{p} - \{p\})) <' p$.

Now, we give the algorithm k-MMG in Figure 2, which computes one of the k-mmg of the given set of trees. We show the correctness of the procedure k-MMG. Let k be a positive number and $S \subseteq \mathcal{T}_\Sigma$ be a finite set. The following Theorem 13 will be proved in the next section.

Theorem 13. Let k be a fixed positive number. For every any finite $S \in \mathcal{T}_\Sigma$, a reduced set of exactly k tree patterns with respect to S can be found in polynomial time with respect to $||S||$ if it exists.

Once the k-MMG procedure finds a reduced set $\bar{p}$ with respect to S that may not be a normal form at Line 1 in Figure 2, the procedure tries to make p a normal form by iteratively tightening it with respect to S, that is, replacing a pattern $p \in \bar{p}$ by more specific pattern $lca(S - L(\bar{p} - \{p\}))$ for every $p \in \bar{p}$.

Lemma 14. Assume that $\sharp\Sigma > k$. Then, after executing lines from Line 1 to Line 7 in Figure2, any $\bar{p} \in (\mathcal{TP}_\Sigma)^k$ output in the line 7 is a normal form with respect to S.

Proof. To prove the lemma, we first show that any applications of tightening to a reduced set $\bar{p} \subseteq \mathcal{TP}_\Sigma$ make $L(\bar{p})$ smaller, but for the resulting set $\bar{q}$, whose language $L(\bar{q})$ still contains S. Assume that for $p \in \bar{p}$, we replace p in $\bar{p}$ by $q = S - L(\bar{p} - \{p\})$. Because $\bar{p}$ is reduced with respect to S, $\emptyset \neq S - L(\bar{p} - \{p\}) \subseteq L(p)$. By Lemma 5 and Lemma 6, we have $q \leq' p$. If let $\bar{q} = (\bar{p} - \{p\}) \cup \{q\}$ be the resulting set, then $S \subseteq L(\bar{q})$. Because $S - L(\bar{p} - \{p\}) \subseteq L(q)$. Therefore, the claim is proved.

Now, assume that we get $p \in \bar{p}$, after at least one application of tightening to p with respect to

S. Then, $p \equiv' lca(S - L(\bar{p} - \{p\}))$. Assume that we further applied tightening to other members in $\bar{p}$ than p, and obtained a set $\bar{q}$. Then, p is contained in both of $\bar{p}$ and $\bar{q}$. By the consideration we first showed, $L(\bar{q} - \{p\}) \subseteq L(\bar{p} - \{p\})$ and $S \subseteq L(\bar{q})$. Thus,

$$S - L(\bar{p} - \{p\}) \subseteq S - L(\bar{q} - \{p\}) \subseteq L(p).$$

From the assumption $p \equiv' lca(S - L(\bar{p} - \{p\}))$ and Lemma 5, we have $p \leq' lca(S - L(\bar{q} - \{p\})) \leq' p$. Hence, further applications of tightening to $p \in \bar{q}$ does not change $\bar{q}$. Hence, the result is proved. □

Lemma 15. Assume that $\sharp\Sigma > k$. If $\bar{p} \in (\mathcal{TP}_\Sigma)^k \backslash (\mathcal{TP}_\Sigma)^{k-1}$ is reduced with respect to S, then there is no $\bar{q} \in (\mathcal{TP}_\Sigma)^{k-1}$ such that $S \subseteq L(\bar{q}) \subseteq L(\bar{p})$.

Proof. Assume that there is $\bar{q} \in (\mathcal{TP}_\Sigma)^{k-1}$ such that $S \subseteq L(\bar{q}) \subseteq L(\bar{p})$. Then, By the compactness with respect to containment of $(\mathcal{TPL}_\Sigma)^k$, there is a mapping h from $\bar{q}$ to $\bar{p}$ such that $L(q) \subseteq L(h(q))$ for every $q \in \bar{q}$. Thus, $S \subseteq L(\bar{q}) \subseteq L(h(\bar{q}))$ for a proper subset $h(\bar{q})$ of $\bar{q}$. This contradicts. □

Lemma 16. Assume that $\sharp\Sigma > k$. If $\bar{p} \in (\mathcal{TP}_\Sigma)^k \backslash (\mathcal{TP}_\Sigma)^{k-1}$ is a normal form with respect to S, then $\bar{p}$ is a k-mmg of S.

Proof. By Lemma 15, there is no $\bar{q} \in (TP)^{k-1}$ such that $S \subseteq L(\bar{q}) \subseteq L(\bar{p})$ since $\bar{p}$ is reduced with respect to S. We next assume that there is $\bar{q} \in (\mathcal{TP}_\Sigma)^k \backslash (\mathcal{TP}_\Sigma)^{k-1}$ such that $S \subseteq L(\bar{q}) \subset L(\bar{p})$.

By the compactness with respect to containment of $(\mathcal{TPL}_\Sigma)^k$, there is a mapping h from $\bar{q}$ to $\bar{p}$ such that $L(q) \subseteq L(h(q))$ for every $q \in \bar{q}$. Since $\bar{p}$ is reduced with respect to S, we can assume that h is one-to-one and $\bar{q}$ is also reduced with respect to S; otherwise, by a similar argument in the proof of Lemma 15, contradiction is immediately derived. From $L(\bar{q}) \subset L(\bar{p})$, $q <' h(q)$ for some $q \in \bar{q}$.

Thus, $L(\bar{q} - \{q\}) \subseteq L(\bar{h(q)} - \{h(q)\})$ implies that $S - L(\bar{h(q)} - \{h(q)\}) \subseteq S - L(\bar{q} - \{q\}) \subseteq L(q)$. If $\bar{p}$ is a normal form, then $h(q) \equiv' S - L(\bar{h(q)} - \{h(q)\})$. Since $A \subseteq B$ implies $lca(A) \leq' lca(B)$, $h(q) \equiv' S - L(\bar{h(q)} - \{h(q)\}) \leq' lca(L(q)) \equiv' q$. However, $q <' h(q)$ from assumption. This contradicts. Hence, there is no $\bar{q} \in (TP)^k$ such that $S \subseteq L(\bar{q}) \subset L(\bar{p})$. □

Theorem 17. *Let k be a positive number and Σ be an alphabet. Assume that $\sharp\Sigma > k$. Then, for any finite set $S \in \mathcal{T}_\Sigma$ of trees, k-mmg of S is well-defined and can be computed in polynomial time with respect to $||S||$.*

Proof. By induction on $k \geq 1$, we show that the procedure M in Figure 2 correctly works. The claim is trivial for $k = 1$ because of the correctness of anti-unification procedure [LMM88]. We assume that $k > 1$ and the m-MMG procedure computes an m-mmg of S for any $m < k$.

If there is a reduced set $\bar{p} \in (\mathcal{TP}_\Sigma)^k \backslash (\mathcal{TP}_\Sigma)^{k-1}$ with respect to S, we can find it by Theorem 13 at Line 1 in Figure 2. Then, Lemma 16 shows the result.

Otherwise, there is no reduced set in $(\mathcal{TP}_\Sigma)^k \backslash (\mathcal{TP}_\Sigma)^{k-1}$. Since k-mmg of S itself is reduced with respect to S, there is no k-mmg of S in $(\mathcal{TP}_\Sigma)^k \backslash (\mathcal{TP}_\Sigma)^{k-1}$. Therefore, $(\mathcal{TPL}_\Sigma)^{k-1}$ contains all of the k-mmg of S. By induction hypothesis, $(k-1)$-mmg of S is well-defined and can be computed by the procedure $(k-1)$-MMG. It is not difficult to see the k-MMG procedure runs in polynomial time in $||S||$. □

The above theorem says that the minimal language problem for $(\mathcal{TPL}_\Sigma)^k$ is polynomial time computable if $\sharp\Sigma > k$. By Lemma 11, the class $(\mathcal{TPL}_\Sigma)^k$ has finite elasticity and by Lemma 10, the membership problem for $(\mathcal{TPL}_\Sigma)^k$ is polynomial time decidable. Thus, by Lemma 9, we obtain the main result of this paper.

Corollary 18. *Let k be a positive number and Σ be an alphabet. Assume that $\sharp\Sigma > k$. Then, the class $(\mathcal{TPL}_\Sigma)^k$ is polynomial time inferable from positive data.*

5 Finding reduced set of tree patterns

In this section we describes the basic ideas and the algorithm to find a reduced pair of tree patterns, which dominates the time complexity of the k-mmg algorithm.

For every $k > 0$, we consider the following search problem, when $\sharp\Sigma > k$,

> Given: a finite set $S \subseteq \mathcal{T}_\Sigma$ of trees.
> Problem: find a reduced set $\bar{p}$ of exactly k tree patterns with respect to S.

The key to an efficient inference algorithm is to realize an algorithm that solves this problem in polynomial time.

There is a simple algorithm to solve the problem by using the anti-unification algorithm as a subprocedure; given S, it searches a set $\bar{p} = \{lca(S_1), \ldots, lca(S_k)\}$ satisfying the condition that any proper subset of $\bar{p}$ does not contains S by enumerating all the partitions $S_1, \ldots, S_k$ of S and checking the condition. However, this simple method does not work efficiently, because the number of all the partitions of S is exponential in $\sharp S$.

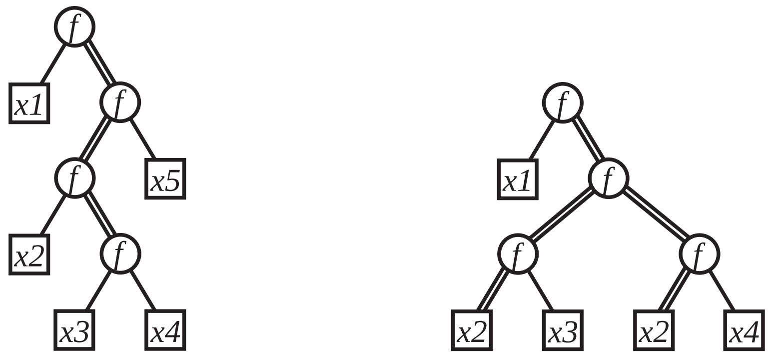
To achieve efficiency, we adopt another approach than checking all the partitions of S; we pay attention only to all the combinations of k distinct trees selected from S, whose number is at most polynomial in $\sharp S$ for fixed k. We relate the following search problem to that of finding a reduced set.

> Given: distinct finite set $Pos, Neg \subseteq \mathcal{T}_\Sigma$ such that $Pos \cap Neg = \emptyset$.
> Problem: find a *maximal, consistent tree pattern* $p\mathcal{TP}_\Sigma$ with Pos and Neg, that is, (1) p is *consistent* with Pos and Neg, i.e., $Pos \subseteq L(p)$ and $Neg \cap L(p) = \emptyset$, and (2) $L(q) \not\subset L(p)$ for any consistent $q \in \mathcal{TP}_\Sigma$ with Pos and Neg.

For instance, $f(x, a)$ and $f(x, x)$ are maximal, consistent tree patterns with $\{f(a, a)\}$ and $\{f(a, b)\}$. We first consider the problem in the case where both of Pos and Neg are singletons.

Lemma 19. (Arimura. et. al.[ASO92]) Let w^+, w^- be constant trees and p be a maximal tree pattern consistent with $\langle w^+, w^-\rangle$. Then, p satisfies either (1) or (2) below. (See Figure 3)

(1) For some node α in p, all leaves in p other than α are labeled by variables, and all internal nodes in p are ancestors of the node α. Moreover, all variables that occur in p are mutually distinct.



$f(x_1, f(f(x_2, f(x_3, x_4)), x_5))$ $\qquad$ $f(x_1, f(f(x_2, x_3), f(x_2, x_4)))$

Figure 3: Possible forms of maximal tree patterns consistent with $\langle w^+, w^- \rangle$

(2) For some distinct leaves α_1, α_2 in p, all leaves in p are labeled by variables, and all internal nodes in p are ancestors of either α_1 or α_2. Moreover, only α_1, α_2 are leaves in p that are labeled by the same variable, and other leaves in p are labeled by mutually distinct variables.

Proof. By Lemma 1, if $w^- \not\leq' p$, then there are two possibilities for p:

1. For some $\alpha \in \mathcal{D}_p$, $w^-(\alpha) \notin X$ and $w^-(\alpha) \neq p(\alpha)$ for $w^-(\alpha), p(\alpha) \in \Sigma$.
2. For some $\alpha, \beta \in \mathcal{D}_p$, $p(\alpha) = p(\beta) \in X$ but $w^-(\alpha) \neq w^-(\beta)$.

In the former case, we construct the tree pattern $q \in \mathcal{TP}_\Sigma$ such that

$$\mathcal{D}_q = \{\gamma i \in \mathcal{D}_p \mid \gamma \in \mathcal{D}_p \text{ is an ancestor of } \alpha \text{ and } 1 \leq i \leq arity(p(\gamma))\}, \text{ and}$$
$$q(\gamma) = \begin{cases} p(\gamma) \in \Sigma & \text{, if } \gamma \text{ is an ancestor of } \alpha, \\ x_\gamma \in X & \text{, otherwise.} \end{cases}$$

In the latter case, we also construct q as

$$\mathcal{D}_q = \{\gamma i \in \mathcal{D}_p \mid \gamma \in \mathcal{D}_p \text{ is an ancestor of either } \alpha \text{ or } \alpha_2, \text{ and } 1 \leq i \leq arity(p(\gamma))\}, \text{ and}$$
$$q(\gamma) = \begin{cases} p(\gamma) \in \Sigma \cup X & \text{, if } \gamma \text{ is an ancestor of either } \alpha \text{ or } \beta, \\ x_\gamma \in X & \text{, otherwise.} \end{cases}$$

In both cases, it is not difficult to see that q is a tree pattern consistent with Pos and Neg that satisfies $p \leq q$. Since p is minimal, consistent tree pattern, p and q coincide. $\square$

Lemma 20. There is an algorithm that, given $w^+, w^- \in \mathcal{T}_\Sigma$, enumerates all the maximal, consistent tree patterns with $\{w^+\}$ and $\{w^-\}$ in polynomial time in n, where $n = |w^+|$. Moreover, the number of the solutions output by the algorithm is $O(n^2)$.

Proof. Any tree pattern p in $MAXTREE(w^+, w^-)$ satisfies conditions (1) or (2) of Lemma 19, and p is a generalization of w^+. Let n be the number of nodes in w^+. Then, the number of generalizations of w^+ satisfying the conditions (1) and (2) are $O(n)$ and $O(n^2)$, respectively. Since the relation $\leq'$ can be determined in linear time, we can easily select members of $MAXTREE(w^+, w^-)$ from such generalizations in polynomial time. $\square$

procedure k-REDUCED;
input: a finite set $S \subseteq \mathcal{T}_\Sigma$ of trees;
output: a reduced set of exactly k tree patterns with respect to S;
begin
1 **for** each combination $w_1, \ldots, w_k \in S$ of k distinct trees **do**
2 **begin**
3 **for** each $1 \leq i \leq k$ **do**
4 **begin**
5 **for** each $1 \leq j \leq k$ such that $j \neq i$ **do**
6 let T_j be the set of all the maximal consistent tree patterns with $\{w_i\}$ and $\{w_j\}$;
7 let $A_i = \{gci(\{q_1, \ldots, q_{k-1}\}) \in \mathcal{TP}_\Sigma \mid (q_1, \ldots, q_{k-1}) \in T_{h_1} \times \ldots \times T_{h_{k-1}}\}$, where $\{h_1, \ldots, h_{k-1}\} = \{1, \ldots, k\} - \{i\}$;
8 **end;**
9 **for** each combination $\bar{p} = \{p_1, \ldots, p_k\}$ where $p_i \in A_i$ for every $1 \leq i \leq k$ **do**
10 **if** $S \subseteq L(\bar{p})$ **then** output $\bar{p}$;
11 **end;**
end;

Figure 4: An algorithm to compute a reduced set

Using the algorithm in Lemma 20, we can efficiently compute the the maximal, consistent tree pattern problem in the case where Neg is not a singleton, because $\mathcal{TPL}_\Sigma$ is closed under intersection.

Lemma 21. There is an algorithm that, given disjoint sets $\{w_0\}, \{w_1, \ldots, w_k\} \subseteq \mathcal{T}_\Sigma$, computes a finite set A that contains all the maximal, consistent tree patterns with $\{w_0\}$ and $\{w_1, \ldots, w_k\}$ in polynomial time in n, where $n = |w_0|$. Moreover, the number of the members in A is $O(n^{2k})$.

Proof. By Lemma 3, $\mathcal{TPL}_\Sigma$ is closed under intersection. Thus, if $p \in \mathcal{TP}_\Sigma$ is a maximal, consistent tree pattern with with $\{w_0\}$ and $\{w_1, \ldots, w_k\}$, then we can represent $L(p)$ as the intersection of languages $L(p_1), \ldots, L(p_k)$, where p_i is defined as maximal, consistent tree patterns with $\{w_0\}$ and $\{w_i\}$ for all $1 \leq i \leq k$. Since the pattern defining the intersection can be computed in polynomial time by 3, the result immediately follows. □

In Figure 4, now we give an algorithm k-REDUCED that computes a reduced set of exactly k tree patterns with respect to the given set S. We prove the correctness of the algorithm.

The proof of Theorem 13: We show that the procedure k-REDUCED in Figure 4 correctly works in polynomial time in $||S||$. Let $S \subseteq \mathcal{T}_\Sigma$ be finite set. Assume that there is a set $\bar{p} = \{p_1, \ldots, p_n\}$ of exactly k tree patterns reduced with respect to S. Because $\bar{p}$ is reduced, the set $S - L(\bar{p} - \{p_i\})$ is not empty for any $1 \leq i \leq k$. Thus, we can choose distinct trees $w_1, \ldots, w_k$ from S as $w_i \in S - L(\bar{p} - \{p_i\})$ for every $1 \leq i \leq k$. Clearly, for every $1 \leq i \leq k$, p_i is consistent with $\{w_i\}$ and $\{w_1, \ldots, w_k\} - \{w_i\}$. Thus, we can take k tree patterns $q_1, \ldots, q_k \in \mathcal{TP}_\Sigma$ where each q_i $(1 \leq i \leq k)$ satisfies (a) $p_i \leq' q_i$, and (b) q_i is a maximal, consistent tree pattern with $\{w_i\}$ and $\{w_1, \ldots, w_k\} - \{w_i\}$. Since $p_i \leq' q_i$ for each $1 \leq i \leq k$, we have $S \subseteq L(\bar{p}) \subseteq L(\bar{q})$. Therefore, it immediately follows that $\bar{p}$ satisfies (c) $S \subseteq L(\bar{q})$, and (d) q_i is a maximal, consistent tree pattern with $\{w_i\}$ and $\{w_1, \ldots, w_k\} - \{w_i\}$ for any $1 \leq i \leq k$. Therefore, if the algorithm select the combination $w_1, \ldots, w_k$ at line 1 in Figure 4, then it can find n $\bar{q}$ in the for-loop from Line 8 to Line 9 after executing lines from Line 3 to Line 8. The obtained set $\bar{q}$ is reduced with respect to S because of the conditions (c) and (d).

Since the number of possible combinations of $\langle t_1, \ldots, t_k \rangle$ is $(\sharp S)^k$ and the maximum number of elements of each T_i $(1 \leq i \leq k)$ is $O(m^2)$, where m is the maximum size of the trees in S, the procedure k-REDUCED runs in time $O(n^{2k^2-k+1}) = n^{O(k^2)}$ with respect to the total size $n = ||S||$ of the input S. □

6 Discussions

We showed that the class of unions of at most k tree pattern languages is consistently and conservatively polynomial time identifiable in the limit from positive data under some restriction on the size of an alphabet. Our algorithm to compute k-mmg of the set of trees can be considered as a natural extension of Plotkin's least generalization algorithm. To prove the correctness of the k-mmg algorithm, the compactness with respect to containment played an important role. For several classes of string pattern languages, Mukouchi [Muk91] proved an interesting series of the results concerning to the compactness with respect to containment.

In this paper, we considered only inference machines working consistently and conservatively with polynomial time updating hypothesis. Pitt [Pit89] proposed another good criterion for efficient identification in the limit; where the total number of implicit errors of prediction is bounded by a polynomial in the size of the unknown hypothesis, not only the time for updating. Unfortunately, our class $(\mathcal{TPL}_\Sigma)^k$ seems not to be identifiable in the limit from positive data under Pitt's criterion.

Lange and Wiehagen showed an interesting method to infer pattern languages from positive data [LW91]. The inference machine developed by them produces a guess in polynomial time which is not guaranteed to be consistent with given examples. However, it should be noticed that their inference machine works in iterative manner [JB81], that is, it produces any guess only from the guess produced last and the current example, and it does not need to remember any other examples read so far. Considering iteratively working inference machines for tree patterns and their unions might be an interesting problem.

Acknowledgments. The first author would like to thank Prof. Setsuo Arikawa for his constant encouragement, and also thank Hiroki Ishizaka and Ayumi Shinohara for fruitful discussions with them. The authors also thank the anonymous referees for their careful comments and suggestions which improved the readability of the paper.

References

[Ang79] D. Angluin. Finding patterns common to a set of strings. In *Proceedings of the 11th Annual Symposium on Theory of Computing*, pages 130–141, 1979.

[Ang80a] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[Ang80b] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[ASO92] H. Arimura, T. Shinohara, and S. Otsuki. Polynomial time inference of unions of two tree pattern languages. *IEICE Trans. Inf. and Syst.*, E75-D(7):426–434, 1992.

[JB81] K.P. Jantke and H-R. Beick. Combining postulates of naturalness in inductive inference. *Elektron. Informationsverarb. Kybern.*, 17:465–484, 1981.

[Kan88] P. C. Kanellakis. Logic programming and parallel complexity. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 547–585. Morgan Kaufmann, 1988.

[LM86] J-L. Lassez and K. Marriott. Explicit representation of terms defined by counter examples. *Journal of Automated Reasoning*, 3:301–317, 1986.

[LMM88] J-L. Lassez, M.J. Maher, and K. Marriott. Unification revisited. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 587–625. Morgan Kaufmann, 1988.

[LW91] S. Lange and R. Wiehagen. Polynomial-time inference of pattern languages. *New Generation Computing*, 8(4):361–370, 1991.

[Muk91] Y. Mukouchi. Characterization of pattern languages. In *Proceedings of the Second Workshop on Algorithmic Learning Theory*, pages 93–104, Tokyo, 1991.

[Pit89] L. Pitt. Inductive inference, DFAs, and computational complexity. In K. P. Jantke, editor, *Proceedings of International Workshop on Analogical and Inductive Inference*, pages 18–44, 1989. Lecture Notes in Computer Science 397.

[Plo70] G. Plotkin. A note on inductive generalization. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

[Rey70] J. Reynolds. Transformational systems and the algebraic structure of atomic formulas. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, volume 5, pages 135–152. Edinburgh University Press, 1970.

[Shi83] T. Shinohara. Inferring unions of two pattern languages. *Bulletin of Informatics and Cybernetics*, 20:83–88, 1983.

[Wri89a] K. Wright. Identification of unions of languages drawn from an identifiable class. In *Proceedings of the 2nd Annual Workshop on Computational Learning Theory*, pages 328–333, 1989.

[Wri89b] K. Wright. *Inductive Inference of Pattern Languages.* PhD thesis, University of Pittsburgh, 1989.