九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Polynomial Time Inference of Unions of Tree Pattern Languages

Arimura, Hiroki
Research Institute of Fundamental Information Science Kyushu University

Shinohara, Takeshi
Research Institute of Fundamental Information Science Kyushu University

Otsuki, Setsuko
Research Institute of Fundamental Information Science Kyushu University

https://hdl.handle.net/2324/3149

出版情報：RIFIS Technical Report. 43, 1991-08-01. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

RIFIS-TR-CS-43

RIFIS Technical Report

Polynomial Time Inference of Unions of Tree Pattern Languages

Hiroki Arimura
Takeshi Shinohara
Setsuko Otsuki

August 1, 1991

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan

E-mail: arim@ai.kyutech.ac.jp Phone: 0948 (29) 7638

POLYNOMIAL TIME INFERENCE OF 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. In this paper we consider the polynomial time inferability from positive data for unions of two tree pattern languages. A tree pattern is a structured pattern known as a term in logic programming and term rewriting systems, and a tree pattern language is the set of all ground instances of a tree pattern. We present a polynomial time algorithm to find a minimal union of two tree pattern languages containing given examples. Our algorithm can be considered as a natural extension of Plotkin's least generalization algorithm, which finds a minimal single tree pattern language. By using this algorithm we can realize a polynomial time inference machine for unions of two tree pattern languages from positive data.

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 two tree pattern languages.

A *tree pattern* is a structured pattern known as a *term* in logic programming and term rewriting systems, and a *tree pattern language* is the set of all ground instances of a tree pattern. Since the class of tree pattern languages has finite thickness, that is, for any ground tree there are only finitely many tree pattern languages containing it, an inference machine that guesses a minimal language explaining examples identifies a target tree pattern from examples in the limit [2]. For example, assume that a rule is represented by a tree pattern and the following ground trees are given as its examples:

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

Then,

$$\{app(X,Y,Z)\}$$

represents a minimal tree pattern language containing all the examples. Such a minimal tree pattern, called the *least generalization* by Plotkin in [8], can be computed by using the *least generalization algorithm* [6, 8] in polynomial time.

In this paper, we will consider inductive inference where examples are taken from *two* tree pattern languages, and pay attention to the problem finding a pair of 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 tree patterns represents a minimal union containing the above examples.

On the other hand, for string pattern languages, the problem whether minimal unions can be computed in polynomial time is open [9, 11]. For one-variable pattern languages, it is shown to be computable in polynomial time [10].

In this paper we study polynomial time inferability of unions of two tree pattern languages from positive data. We first prove that the containment problem for unions of n tree pattern languages is decidable in polynomial time if n is less than the number of symbols of alphabet.

Wright [11] proved that a minimal unions containing a sample can be computed in polynomial time for one-variable pattern languages. However we can not apply his method directly to unions of tree pattern languages. Because the number of generalizations of a given finite sample may be exponential for tree patterns, while it is at most polynomial for one-variable patterns.

We show that it is not necessary to check all partitions of a sample, but it is sufficient to find a pair of tree patterns reduced with respect to the sample and tighten it. A pair of tree patterns is said to be *reduced with respect to* a set S of ground trees if S can not contained in either of their languages, but is contained in the union. We can find such a reduced pair in polynomial time of the size of S , by using an algorithm that receives a pair $\langle w_+, w_- \rangle$ of ground trees taken from S and enumerates maximal tree patterns whose languages include w_+ but exclude w_-. The point to guarantee polynomial time complexity of our algorithm is that the number of such pairs $\langle w_+, w_- \rangle$ is proportional to the square of the size of S and the number of maximal tree patterns consistent with $\langle w_+, w_- \rangle$ is at most the square of the size of w_+.

By a similar discussion for the classes with finite thickness, we can easily show that a machine producing a minimal pair of tree patterns as its guess can identify the class of unions in the limit. Hence, we prove that the class of unions of two tree pattern languages is polynomial time inferable from positive data.

2 Preliminaries

We start with basic definitions on tree patterns and give a brief review of the articles on inductive inference from positive data.

2.1 Tree Pattern Languages

Σ is a finite alphabet associated with a mapping *arity* from Σ to nonnegative integers, whose elements are called *functors*. We assume that Σ contains at least one functor with arity 0. V is a countable set of symbols disjoint from Σ, whose elements are called *variables*.

A *tree pattern* is defined recursively as follows: t is a tree pattern on $\Sigma \cup V$ if t is (1) a functor a with $arity(a) = 0$, (2) a variable, or (3) an ordered tree $f(t_1, \ldots, t_n)$, where f is an n-ary functor labeling the root node and $t_1, \ldots, t_n$ are tree patterns on $\Sigma \cup V$.

A tree pattern containing no variables is called a *constant tree*. We denote by $\mathcal{TP}(\Sigma)$ the set of all tree patterns on $\Sigma \cup V$ and by $\mathcal{T}(\Sigma)$ the set of all constant tree patterns on Σ. Note that our definition of tree patterns is slightly different from that of Ko et al. [5]. In their definition, the arity of a functor is variable.

A *substitution* is a homomorphism θ from tree patterns to themselves such that $\theta(a) = a$ for each 0-ary functor $a \in \Sigma$. We define a *matching relation* $\leq'$ on tree patterns as $p \leq' q$, if $p = \theta(q)$ for some substitution θ. The *language defined by a tree pattern* p is the set

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

A subset L of $\mathcal{T}(\Sigma)$ is a *tree pattern language* on Σ if there is a tree pattern $p \in \mathcal{TP}(\Sigma)$ such that $L = L(p)$. We denote by $\mathcal{TPL}(\Sigma)$ the class of tree pattern languages for a given alphabet Σ.

Example 1. Suppose that $\Sigma = \{a, b, f\}$ and $arity(a) = 0, arity(b) = 0$ and $arity(f) = 2$. Then, $f(x, x)$ is a tree pattern, and constant trees $f(a, a)$ and $f(f(a, a), f(a, a))$ are contained in the tree pattern language $L(f(x, x))$, but a constant tree $f(a, b)$ is not contained in $L(f(x, x))$.

2.2 Inductive Inference from Positive Data

First, we give a basic definitions on inductive inference according to Gold [4].

An *indexed family of recursive languages* is a class of languages $\mathbf{C} = \{L_1, L_2, L_3, \ldots\}$ such that there is an effective procedure to decide whether $w \in L_i$ given a word w and an index i. For the class $\mathcal{TPL}(\Sigma)$ of tree pattern languages, a word is a constant tree and an index of $L(p)$ is a tree pattern p. For a class $\mathbf{C}$ and a nonnegative integer m, we define the class $\mathbf{C}^m$ of unions of m languages in $\mathbf{C}$ as $\mathbf{C}^m = \{L_1 \cup \ldots \cup L_m \mid L_1, \ldots, L_m \in \mathbf{C}\}$. A *positive presentation* of L is an infinite sequence $w_1, w_2, \ldots$ such that $\{w_i \mid i \geq 1\} = L$. A *sample* is a nonempty finite set of words.

An *inference machine* is an effective procedure M that requests a word and produces a guess from time to time. Let $\sigma = w_1, w_2, \ldots$ be a positive presentation. When M makes the i-th request, a word w_i is added to the sample. Then, M reads the current sample and adds a guess g_i to the sequence of guesses. We say that M *on input* σ *converges to* g if there exists a positive integer N such that $g_i = g$ for every $i \geq N$.

An inference machine M is said to be *consistent* if it always produces a guess g_i consistent with the current sample, that is, $\{w_1, \ldots, w_i\} \subseteq L_{g_i}$ for every $i > 0$. M is said to be *conservative* if it continues to produce the same guess while the guess is consistent with the sample.

A class of languages $\mathbf{C} = \{L_1, L_2, \ldots\}$ is said to be *inferable from positive data* if there exists an inference machine M such that M on input σ converges to g with $L_g = L_i$ for any index i and any positive presentation σ of L_i.

A class $\mathbf{C}$ has *finite thickness*, called Condition 3 by Angluin in [3], if the set $\{L \in \mathbf{C} \mid w \in L\}$ is finite for any word w. Angluin showed that if a class $\mathbf{C}$ has finite thickness, then $\mathbf{C}$ is inferable from positive data [3]. Using this condition she proved that the class of string pattern languages is inferable from positive data.

Wright extended her result to unions of languages [10]. A class $\mathbf{C}$ has *infinite elasticity* if there exist two infinite sequences $w_0, w_1, \ldots$ of words and $L_1, L_2, \ldots$ of languages in $\mathbf{C}$ such that $w_i \notin L_i$ and $w_j \in L_i$ if $j < i$. $\mathbf{C}$ has *finite elasticity* if $\mathbf{C}$ does not have infinite elasticity. Clearly, if a class $\mathbf{C}$ has finite thickness then $\mathbf{C}$ has finite elasticity.

Theorem 1. ([10]) If a class **C** has finite elasticity, then **C** is inferable from positive data.

He also proved that this property is closed under union of languages. Since the class $\mathcal{PL}$ of string pattern languages has finite thickness and finite elasticity is closed under union, the class $\mathcal{PL}^m$ of unions of at most m string pattern languages is inferable from positive data [10].

Next we consider the polynomial time inferability.

Definition 1. A class of languages **C** is said to be *polynomial time inferable from positive data* if there exists an inference machine M that infers **C** consistently and conservatively from positive data, and it computes the guess g_i in polynomial time with respect to $||S_i||$ for every stage $i > 0$, where $S_i = \{w_1, \ldots, w_i\}$ and $||S_i||$ is the size of the sample S_i as an expression.

Angluin [1] showed the following sufficient condition for polynomial time inference from positive data.

Proposition 2. (Angluin [1]) If a class **C** has finite thickness and both of membership and MINL calculations for **C**, that is,

$$\begin{aligned} \mathrm{MINL}(S) \quad = \quad & \text{"Given a finite set } S \text{ of words, find an index} \\ & g \text{ such that } S \subseteq L_g, \text{ and } L_i \subsetneq L_g \text{ implies} \\ & S \not\subseteq L_i \text{ for any } i\text{"}. \end{aligned}$$

are computable in polynomial time with respect to $||S||$, then the following procedure M infers **C** from positive data in polynomial time, where $||S||$ is the size of S as an expression.

Algorithm 1. (inference machine M)

procedure M;
    input: an positive presentation $w_1, w_2, \ldots$;
    output: an infinite sequence $g_1, g_2, \ldots$ of guesses;
begin
$g_0 := none$; $S := \emptyset$; $i := 0$; { stage 0 }
repeat { stage i }
    $i := i + 1$;
    $S := S \cup \{w_i\}$;
    if $w_i \notin L_{g_{i-1}}$ **then** $g_i := \mathrm{MINL}(S)$ **else** $g_i := g_{i-1}$;
    output g_i;
forever;
end.

A fundamental procedure used in the above algorithm is that for MINL calculation. For subclasses of single string pattern languages, polynomial time algorithms for MINL calculation are proposed.

We can easily see that the class $\mathcal{TPL}(\Sigma)$ of tree pattern languages has finite thickness [1, 3]. Thus, the class $\mathcal{TPL}(\Sigma)^m$ of unions of m tree pattern languages has finite elasticity, but does not have finite thickness for any $m > 1$.

Let us consider how the inference machine M behaves for a class **C** with finite elasticity. Let σ be a positive presentation of a language L_k in **C**. Let $g_0, g_1, \ldots$ be a subsequence

of distinct guesses produced by M on σ and $w_0, w_1, \ldots$ be an input data that cause these changes of guesses. If M on σ does not converge, then these two sequences are infinite and 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, since M outputs an index of a minimal language L_{g_i} containing a sample S_i at any stage $i \geq N$, $L_k \supseteq L_{g_i}$. Hence, for a class $\mathbf{C}$ with finite elasticity, if both of membership and MINL calculations are computable in polynomial time, then M in algorithm 1 infers $\mathbf{C}$ in polynomial time from positive data.

3 Inference of Unions of Tree Pattern Languages

In this section we describe the main procedure in our algorithm to infer a union of two tree pattern languages from positive data in polynomial time.

3.1 Compactness with respect to Containment

We first observe a basic property of unions of tree pattern languages. This property, *compactness with respect to containment*, plays an important role to guarantee the correctness of our algorithm.

Definition 2. Let $\mathbf{C}$ be a class of languages and $m > 0$. The class $\mathbf{C}^m$ of unions is *compact with respect to containment* if for every $L, L_1, \ldots, L_m \in \mathbf{C}$,

$$L \subseteq L_1 \cup \ldots \cup L_m \quad \Rightarrow \quad L \subseteq L_i \text{ for some } 1 \leq i \leq m.$$

We denote by $\sharp S$ the number of elements in a set S.

Theorem 3. *The class $\mathcal{TPL}(\Sigma)^m$ of unions of m tree pattern languages is compact with respect to containment if $\sharp\Sigma > m$.*

Proof. Suppose that $\sharp\Sigma = s > m$ and $L(p) \subseteq L(q_1) \cup \ldots \cup L(q_m)$ for $p, q_1, \ldots, q_m \in \mathcal{TP}(\Sigma)$. Assume here that Σ consists of s functors $f_1, \ldots, f_s$ and p contains k distinct variables $x_1, \ldots, x_k$. We choose a set $\tau_0 = \{t_1, \ldots, t_s\} \subseteq \mathcal{T}(\Sigma)$ of s constant trees such that the root of each constant tree t_i is labeled by f_i for each $1 \leq i \leq s$.

Let Θ_0 be the set of all substitutions from variables in p to τ_0, that is, $\Theta_0 = \{\theta \mid \theta(x_i) \in \tau_0 \text{ for } 1 \leq i \leq k\}$. For a set of substitutions Δ, we define the direct image $\Delta(p)$ of a pattern p as $\Delta(p) = \{\theta(p) \mid \theta \in \Delta\}$. Clearly, $\Theta_0(p)$ is a finite subset of $L(p)$ with s^k elements. Since $\Theta_0(p) \subseteq L(p)$, $\Theta_0(p) \subseteq L(q_1) \cup \ldots \cup L(q_m)$. Thus, for some $1 \leq i \leq n$, one language $L(q_i)$ must contain at least $1/m$ of $\Theta_0(p)$. Let $\Theta(p) = \Theta_0(p) \cap L(q_i)$. Since $s > m$, $\sharp\Theta(p) \geq s^k/m > s^{k-1}$. Now, consider the following condition.

Condition 1. Θ satisfies either (1) there is some $1 \leq i \leq k$ such that $\theta(x_i) = \theta'(x_i)$ for every $\theta, \theta' \in \Theta$, or (2) there is some $1 \leq i, j \leq k$ such that $\theta(x_i) = \theta(x_j)$ for every $\theta \in \Theta$. (Table 1)

Claim 1. If a subset Θ of Θ_0 satisfies Condition 1, then $\sharp\Theta(p) \leq s^{k-1}$.

Proof. If we fix the i-th variable as $\theta(x_i) = t$ for some tree t, then there are at most s^{k-1} distinct assignments for other variables. A similar combinatorial argument shows the result.

i	$\theta_i(x_1)$	$\theta_i(x_2)$	$\theta_i(x_3)$	$\theta_i(x_4)$
1	a	b	c	**d**
2	b	**c**	a	**d**
3	d	**c**	b	a

i	$\theta_i(x_1)$	$\theta_i(x_2)$	$\theta_i(x_3)$	$\theta_i(x_4)$
1	a	b	c	d
2	b	c	d	a
3	c	**a**	**a**	b

Table 1: Two examples of Θ violating Condition 1 in the proof of Theorem 3

From this claim, $\Theta(p)$ must not satisfy Condition 1. Thus, we can show that $lca(\Theta(p)) = p$ by using Boomerang Lemma [6], where $lca(A)$ is the least generalization of a set A of constant trees. Since $\Theta(p) \subseteq L(q_i)$, $p = lca(\Theta(p)) \leq' q_i$. Hence it is followed by the result. □

The condition in Theorem 3 that the alphabet Σ has more than m functors is necessary for the class $\mathcal{TPL}(\Sigma)^m$ $(m > 0)$. For instance, suppose that $\Sigma = \{a, f\}$, $arity(a) = 0$ and $arity(f) = 2$. Then, $L(x)$ is contained in the union $L(a) \cup L(f(x_1, x_2))$, but $L(x)$ is contained in neither $L(a)$ nor $L(f(x_1, x_2))$.

This theorem also shows that the containment problem for the class $\mathcal{TPL}(\Sigma)^m$ of unions of m tree pattern languages, that is, given a pair of multisets $\{M_1, \ldots, M_m\}$, $\{L_1, \ldots, L_m\}$ of m tree patterns, the problem whether $M_1 \cup \ldots \cup M_m \subseteq L_1 \cup \ldots \cup L_m$ can be computable in polynomial time.

3.2 Finding a Minimal Union

We describe here a MINL algorithm based on the technique used in [11]. First we introduce the notion of reduced unions. For a multiset of tree patterns $P = \{p_1, \ldots, p_m\}$, we denote by $L(P)$ the union $L(p_1) \cup \cdots \cup L(p_m)$ defined by P.

Definition 3. A multiset of tree patterns $P = \{p_1, \ldots, p_m\}$ is *reduced* with respect to a sample S if (1) the union $L(P)$ contains S, and (2) for any pattern $p_i \in P$, $L(P - \{p_i\})$ does not contains S.

For instance, a pair $\{f(a, x), f(x, b)\}$ of tree patterns is reduced with respect to $\{f(a, a), f(a, b), f(b, b)\}$, but $\{f(a, x), f(x, y)\}$ is not. Note that for any tree pattern p and a sample S a multiset $\{p, p\}$ is not reduced with respect to S.

The MINL algorithm consists of the following steps. Let Σ be an alphabet with at least three functors and S be a sample.

Algorithm 2.
1. Find a reduced pair $\langle p_0, q_0 \rangle$ of tree patterns with respect to S.
2. If it is found, compute the least generalizations p of $S - L(q_0)$ and q of $S - L(p)$. Then output $\langle p, q \rangle$.
3. If there is not such a $\langle p_0, q_0 \rangle$, compute the least generalization u of S. Then, output $\langle u, u \rangle$.

We first compute a reduced pair $\langle p_0, q_0 \rangle$ with respect to S if it exists. This algorithm runs in polynomial time with respect to the size of S. Note that we can compute the minimal single tree pattern of a sample by using the least generalization algorithm [6, 8] in polynomial time.

Suppose here that such a pair is found. By similar arguments in proofs of Lemma 2.5 and Lemma 2.8 in [11], we can obtain the following lemma.

Lemma 4. Suppose that $\sharp\Sigma \geq 3$ and a pair $\langle p_0, q_0 \rangle$ of tree patterns is reduced with respect to S. Let p and q be the least generalizations of $S - L(q_0)$ and $S - L(p)$, respectively. Then, the union $L(p) \cup L(q)$ is a minimal union containing S.

Proof. Let p be the least generalization of $S - L(q_0)$. Then, both of p_0 and p contain $S - L(q_0)$. Since $L(p)$ is the least tree pattern language containing $S - L(q_0)$, $L(p) \subseteq L(p_0)$. Let q be the least generalization of $S - L(p) \neq \emptyset$. Similarly, we have $L(q) \subseteq L(q_0)$.

Assume that $S \subseteq L(p') \cup L(q') \subseteq L(p) \cup L(q)$. By the compactness with respect to containment, there are two cases: (1) one of $L(p)$ and $L(q)$ contains both of $L(p')$ and $L(q')$; (2) each of $L(p)$ and $L(q)$ contains exactly one of $L(p')$ and $L(q')$. Since the case (1) contradicts reducedness of $L(p) \cup L(q)$, we assume that $L(p') \subseteq L(p)$ and $L(q') \subseteq L(q)$ without loss of generality. Thus, we have $S - L(q_0) \subseteq S - L(q) \subseteq S - L(q') \subseteq L(p') \subseteq L(p)$. Therefore, both of $L(p)$ and $L(p')$ contain $S - L(q_0)$. This shows $L(p) = L(p')$. Similarly, we have $L(q) = L(q')$. □

Lemma 5. Let S be a sample and u be the least generalization of S. If $S \subseteq L(p) \cup L(q) \subsetneq L(u)$, then $\langle p, q \rangle$ is reduced with respect to S.

Lemma 4 and Lemma 5 give a necessary and sufficient condition for finding a minimal union containing S. If we find one of reduced pairs with respect to S, we can get a minimal union $L(p) \cup L(q)$ containing S in polynomial time (Step 2 in Algorithm 2). Otherwise, no smaller union than $L(u)$ can contain S.

As we will see in the next section, Step 1 of Algorithm 2 is computable in polynomial time. Therefore, Algorithm 2 correctly computes MINL for unions of two tree pattern languages in polynomial time.

The class of unions of two tree patterns has finite elasticity. Thus, by the discussion in Section 2.2, we obtain the main result of this paper.

Theorem 6. *If $\sharp\Sigma \geq 3$, the class $\mathcal{TPL}(\Sigma)^2$ of unions of two tree pattern languages is polynomial time inferable from positive data.*

4 Computing a Reduced Pair

In this section we describes the basic idea and the algorithm to find a reduced pair of tree patterns, which plays an important role to realize our efficient inference machine.

Let S be a sample. The following is a straightforward way to find a pair of tree patterns reduced with respect to S

Algorithm 3.

procedure;
input: a sample S;
output: a pair of tree patterns reduced with respect to S if exists;
begin
for each partition $\{S_1, S_2\}$ of S **do begin**
Let p_1 be the least generalization of S_1;
Let p_2 be the least generalization of S_2;
if $S \not\subseteq L(p_1)$ and $S \not\subseteq L(p_2)$ **then**
output $\langle p, q \rangle$ and **exit**;
end;
end.

However, this method does not work efficiently because there are exponentially many different partitions of a sample S. On the contrast, our algorithm can compute a pair reduced with respect to S in polynomial time. The algorithm is based on the fact that a candidate tree pattern conposing a reduced pair with respect to S can be found out of maximal tree patterns whose languages include w_+ and exclude w_- for some $w_+, w_- \in S$.

Definition 4. Let w_+, w_- be constant trees. A tree pattern p is *consistent with* a pair $\langle w_+, w_- \rangle$ if $w_+ \in L(p)$ and $w_- \notin L(p)$. A consistent tree pattern p is a *maximal tree pattern consistent with* $\langle w_+, w_- \rangle$ if $q \not\leq' p$ for every tree pattern q consistent with $\langle w_+, w_- \rangle$. We denote by $MAXTREE(w_+, w_-)$ the set of all maximal tree patterns consistent with $\langle w_+, w_- \rangle$.

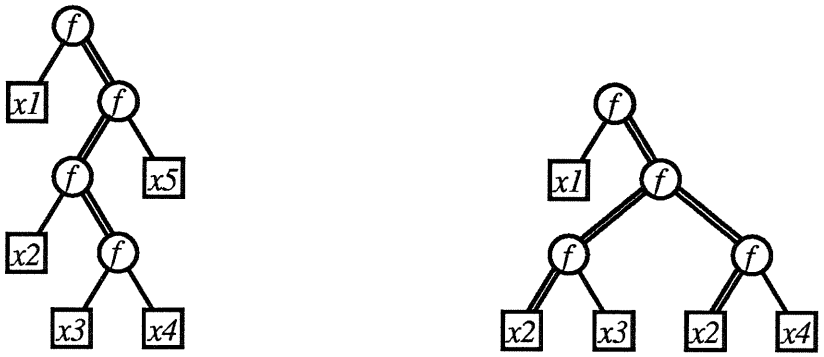
Example 2. $f(x, a)$ and $f(x, x)$ are maximal tree patterns consistent with a pair $\langle w_+, w_- \rangle = \langle f(a, a), f(a, b) \rangle$.

Algorithm 4.

procedure;
input: a sample S;
output: a pair of tree patterns reduced with respect to S if exists;
begin
for each $w_+, w_- \in S$ **do**
for each $q \in MAXTREE(w_+, w_-)$ **do begin**
Let p be the least generalization of $S - L(q)$;
if $\langle p, q \rangle$ is reduced with respect to S **then**
output $\langle p, q \rangle$ and **exit**;
end;
end.

Lemma 7. Let p be a maximal tree pattern consistent with $\langle w_+, w_- \rangle$. Then, either (1) or (2) below holds. (See Figure 1)

1. All but at most one leaves of p are mutually distinct variables, and the other nodes are functors on the same path.



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

Figure 1: Possible forms of maximal tree patterns consistent with $\langle w_+, w_- \rangle$

2. All leaves of p are variables, and just two of them, say α and β, are the same variable, and all internal nodes are on either paths from the root to α or β.

Lemma 8. Given a pair $\langle w_+, w_- \rangle$ of constant trees, the set $MAXTREE(w_+, w_-)$ can be computable in polynomial time of n and $\sharp(MAXTREE(w_+, w_-)) = O(n^2)$, where n is the number of nodes in w_+.

Proof. Any tree pattern p in $MAXTREE(w_+, w_-)$ satisfies conditions (1) or (2) of Lemma 7, 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 matching relation can be determined in linear time, we can easily select members of $MAXTREE(w_+, w_-)$ from such generalizations in polynomial time. □

By the following lemmas, Algorithm 4 finds a pair reduced with respect to a sample in polynomial time.

Lemma 9. Assume that there is at least one pair reduced with respect to S. Then, there exists two constant trees w_+, w_- and a tree pattern q such that $q \in MAXTREE(w_+, w_-)$ and $\langle p, q \rangle$ is reduced with respect to S, where p is the least generalization of $S - L(q)$.

Proof. Assume that there is a pair $\langle p_0, q_0 \rangle$ reduced with respect to S. Let q be a maximal tree pattern such that $L(q_0) \subseteq L(q)$ and $S - L(q) \neq \emptyset$. Note that the existence of q_0 ensures that of such q. For any tree pattern r, $S \cap L(r) \neq \emptyset$ and $S - L(r) \neq \emptyset$ iff r is consistent with $\langle w_+, w_- \rangle$ for some w_+, w_- in S. Thus, $q \in MAXTREE(w_+, w_-)$. Let p be the least generalization of $S - L(q)$. Then, we have $L(p) \subseteq L(p_0)$ from $L(q_0) \subseteq L(q)$. Since $S \not\subseteq L(p_0)$, $S \not\subseteq L(p)$. □

Lemma 10. For any sample S, Algorithm 4 runs in polynoimial time with respect to $||S||$, and it finds a reduced pair with respect to S if exists.

5 Discussions

In this paper, we have shown that the class of unions of two tree pattern languages is polynomial time inferable from positive data. Our algorithm can be considered as a natural extension of Plotkin's least generalization algorithm.

Some of logic program synthesis systems [7] use a naive algorithm, such as Algorithm 3, to compute a minimal union explaining given examples by considering all partitions. However, it is not efficient because they need check exponentially many distinct partitions in the worst case. Since our algorithm computes one of such minimal unions in polynomial time, it can be used as a polynomial time generalization procedure in some cases.

References

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

[2] Angluin, D., Finding common patterns to a set of strings, *Information and Control* **21**, 46–62, 1980.

[3] Angluin, D., Inductive Inference of Formal Languages from Positive Data, *Information and Control* **45**, 117–135, 1980.

[4] Gold, E.M., Languages Identification in the Limit, *Information and Control* **10**, 447–474, 1967.

[5] Ko, K-I., Marron, A. and Tzeng, W-G., Learning String Patterns and Tree Patterns from Examples, In *Proceedings of third annual workshop on Computational Learning Theory*, pages 384–391, 1990.

[6] Lassez, J-L., Maher, M.J. and Marriott, K., Unification Revisited, In Minker, J., editor, *Foundations of Deductive Databases and Logic Programming*, pages 149–176. Morgan Kaufmann, 1988.

[7] Muggleton, S., Machine Invention of first-order predicates by inverting resolution, In *Proceedings of the Fifth International Conference on Machine Learning*, pages 339–352, 1988.

[8] Plotkin, G., A Note on Inductive Generalization, In Meltzer, B. and Mitchie, D., editors, *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

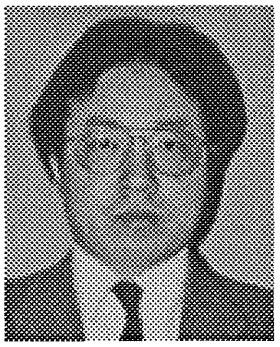
[9] Shinohara, T., Inferring Unions of Two Pattern languages, *Bulletin of Informatics and Cybernetics* **20**, 83–88, 1983.

[10] Wright, K., 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.

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

About the Authors

Hiroki Arimura (有村　博紀) was born in Fukuoka on June 7, 1965. He received the B.S. degree in 1988 in Physics and the M.S. degree in 1990 in Information Systems from Kyusyu University. Presently, he is an Assistant at Department of Artificial Intelligence, Kyushu Institute of Technology, Iizuka. His research interests are in logic programming and Inductive Inference.



Takeshi Shinohara (篠原　武) was born in Fukuoka on January 23, 1955. He received the B.S. in 1980 from Kyoto University, and the M.S. degree and the Dr. Sci. from Kyushu University in 1982, 1986, respectively. Currently, he is an Associate Professor of Department of Artificial Intelligence, Kyushu Institute of Technology. His present interests include information retrieval, string pattern matching algorithms and computational learning theory.

Setsuko Otsuki (大槻　説乎) was born in Tokushima on july 8, 1932. She gradudated from Department of Physics in 1955, and the Dr. Engineering. degree in 1971 in Department of Engneering from Kyushu University. Presently, she is Professor of Department of Artificial Intelligence, Kyushu Institute of Technology. Her research interests include intelligent tutoring systems, knowledge information processing and natural language understanding.

Research Institute of Fundamental Information Scinece, Kyushu University 33, Fukuoka 812, Japan.