Proc. The fifth ACM Workshop on Computational Learning Theory (COLT'92), 136-143, Aug. 1992

Polynomial Time Inference of A Subclass of Context-free Transformations

Hiroki ARIMURA[†] **Hiroki ISHIZAKA**[‡ *] **Takeshi SHINOHARA**[†]

†Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820, JAPAN
arim@ai.kyutech.ac.jp
shino@ai.kyutech.ac.jp

‡Institute for New Generation
Computer Technology
4-28, Mita 1-chome
Minato-ku, Tokyo 108, JAPAN
ishizaka@icot.or.jp

Abstract

This paper deals with a class of Prolog programs, called context-free term transformations (CFT). We present a polynomial time algorithm to identify a subclass of CFT, whose program consists of at most two clauses, from positive data; The algorithm uses 2-mmg (2-minimal multiple generalization) algorithm, which is a natural extension of Plotkin's least generalization algorithm, to reconstruct the pair of heads of the unknown program. Using this algorithm, we show the consistent and conservative polynomial time identifiability of the class of tree languages defined by CFTFB_{uniq} together with tree languages defined by pairs of two tree patterns, both of which are proper subclasses of CFT, in the limit from positive data.

1 Introduction

The problem considered in this paper is, given an infinite sequence of facts which are true in the unknown model, to identify a Prolog program P that defines the unknown model M in the limit. We deal with the class CFTFB_{uniq} of context-free term transformations with a flat base case and the class TP^2 of pairs of two clauses with no bodies. We show the following result:

*Currently works at International Institute for Advanced Study of Social Information Science, FUJITSU LABORATORIES LTD, 140 Miyamoto, Numazu 410-03, Japan. hiro@iias.flab.fujitsu.co.jp

(appeared in Proc. COLT'92, Pittsburgh,July 1992)

Theorem. Let p the only predicate symbol of fixed arity $m \geq 0$ and Σ be the alphabet with $\sharp\Sigma > 2$. Then, least Herbrand models defined by CFTFB_{uniq} together with those defined by TP^2 are polynomial time identifiable in the limit from positive data.

As the inference criterion, we use a criterion proposed by Angluin in the paper [Ang79] published in 1979: *consistent and conservative identification in the limit from positive data with polynomial time of updating conjectures.* In this criterion, identification is an infinite process of receiving a example taken from the unknown language and guessing a conjecture in polynomial time in the total size of the current sample, where the current sample is the set of examples the algorithm received; Moreover, the inference algorithm must output only a conjecture that explains the current sample, and must not change a conjecture whenever it explains the current sample.

From a practical point of view, as in the area of inductive logic programming [Ish88, Mug88, Sha81], inductive inference using this criterion has three major advantages as follows:

First, an inference algorithm which satisfy this criterion requires only positive examples of the unknown languages; for a human teacher, it is considered to be a difficult task to give negative examples of the unknown language to the inference algorithm. Secondly, the inference algorithm quickly responds to an input. Thirdly, the inference algorithm outputs one of the "best-fit" conjectures to the current sample; a consistent and conservative inference algorithm outputs a conjecture that describes a minimal language containing the current examples within a target class of languages [Wri89b].

Moreover, the condition of consistency and conservativeness gives the other advantage; For identification with polynomial time updating conjectures, Pitt pointed out in [Pit89] the problem that if we have any exponential time inference algorithm which identifies a class, then we can obtain an algorithm that runs only with polynomial time updating conjectures by postponing to output a conjecture until they have enough size of examples; However, if we require both of consistency and conservativeness, then we can exclude at least such a kind of essentially exponential time algorithms.

For reasons explained above, we adopt consistent and conservative identification in the limit from positive data with polynomial time of updating hypotheses as our criterion. In this paper, we does not consider any other criteria of efficient learning: PAC-learning, MAT-learning, and another criterion of polynomial time identification in the limit proposed by Pitt [Pit89].

This paper deals with a subclass CFTFB_{uniq} of context-free term transformation. Context-free term transformations were originally introduced by Shapiro in the study on MIS [Sha81], and the class is a subclass of Linear Prolog shown to be identifiable in the limit from positive data, but not effectively. The class CFTFB_{uniq} is a very restricted one, whose program has at most two clauses, but contains several natural small Prolog programs such as append/3. The class is incomparable to the class of tree languages defined by reversible skeletal tree automata, which is shown to be identifiable in the limit from positive data with polynomial time updating by Sakakibara [Sak89]. Further, the technique used in this paper is different from one used in Sakakibara's [Ang82, Sak89]. Our inference algorithm uses an algorithm called 2-mmg [ASO91], which is a natural extension of Plotkin's least generalization (anti-unification) algorithm, to reconstruct the heads of the unknown Prolog program from positive examples. The 2-mmgalgorithm, given a finite set S of atoms, computes a pair of atoms that defines a minimal language containing S within tree languages defined by the class TP^2 of pairs of tree patterns. For example, suppose that the following ground atoms are given as examples of the unknown context-free transformation:

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

Then, the pair

$$\{app([],X,X),\ app([X|Y],Z,[X|W])\}$$

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

Once the inference algorithm have such a pair as a candidate of heads, it searches a context-free transformation with the given pair as heads which is consistent with examples by adding a goal to the body of the pair and outputs the following program

$$app([],X,X]),$$
$$app([X|Y],Z,[X|W]){:-}app(Y,Z,W),$$

as the conjecture. If the inference algorithm can not find any consistent hypothesis with examples whose heads are the given pair, then it simply output the pair of atoms as an approximation of the model by unions of two tree pattern languages.

After proving some preliminary results for CFTFB_{uniq}, we show that the class of least Herbrand models defined by CFTFB_{uniq} together with those defined by TP^2 are polynomial time identifiable in the limit from positive data.

2 Preliminaries

2.1 Tree pattern languages and least generalizations

We refer to both of a term and an atom as *tree pattern*, to a tree pattern contains no variables as a *ground* tree pattern. By $\mathcal{T}_\Sigma$ we denote the set of all ground terms over Σ. 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 size $|t|$ of a tree pattern t is the total number of occurrences of variable symbols, function symbols and predicate symbols in t.

For a tree pattern t, by $L(t)$ we denote the set of all ground instances of a tree pattern t. A set $L \subseteq \mathcal{T}_\Sigma$ is called *tree pattern language* if $L = L(t)$ for some tree pattern t. Let p, q be tree patterns.

Lemma 1 ([LMM88]) $L(p) \subseteq L(q)$ iff $p \leq' q$.

Let $m > 0$ be any fixed number. The *union defined by a set* $\{t_1,\ldots,t_m\}$ *of at most* m *tree patterns*, denoted by $L(\{t_1,\ldots,t_m\})$, is a union $L(t_1)\cup\cdots\cup L(t_m)$. We refer to the class of unions of at most m tree pattern languages as $(\mathcal{TPL}_\Sigma)^m$ and to the class of sets of at most m tree patterns as $(\mathrm{TP}_\Sigma)^m$. We may omit the subscript Σ if it is clear from context. The class $(\mathcal{TPL}_\Sigma)^m$ is *compact with respect to containment* if for any $L \in \mathcal{TPL}_\Sigma$ and any union $L_1\cup\cdots\cup L_m \in (\mathcal{TPL}_\Sigma)^m$,

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

The following proposition concerning containment between tree pattern languages is frequently used in this paper. For a set S, we denote by $\sharp S$ the number of elements of S.

Proposition 2 (Arimura et. al. [ASO91]) Let Σ be an alphabet. For any $m > 0$, if $\sharp\Sigma > m$, then the class $(TPL_\Sigma)^m$ is compact with respect to containment.

A pair $\{p_0, p_1\}$ is *reduced* if $L(p_0) \subseteq L(p_1)$ and $L(p_0) \supseteq L(p_1)$. From the compactness w.r.t. containment, the following lemma is derived. By the compactness w.r.t. containment, we can extend Lemma 1 as follows.

Lemma 3 Let Σ be alphabet with $\sharp\Sigma > 2$ and $\{p_0, p_1\}, \{q_0, q_1\} \in \mathrm{TP}^2$. Assume that both sets are reduced. Then, $L(\{p_0, p_1\}) = L(\{q_0, q_1\})$ iff $\{p_0, p_1\} \equiv' \{p_0, p_1\}$.

Proof. If part is trivial. We show only-if part. If $L(\{p_0, p_1\}) = L(\{q_0, q_1\})$, then there are $4 \times 4 = 16$ possible combinations by the compactness shown above. However, since the relation $\subseteq$ satisfies transitivity it is easily considered by simple case analysis that there are two major cases. Let $i, j, i', j' \in \{0, 1\}$. One is that $L(p_0) \subseteq L(p_1)$ for some $i \neq j$; it is impossible because they are reduced. Another is that $L(p_i) = L(q'_i)$ and $L(p_j) = L(q'_j)$ for some $i \neq j$ and $i' \neq j'$; This implies $\{p_0, p_1\} \equiv' \{q_0, q_1\}$. □

A tree pattern t is a *generalization* of a tree pattern s if $s \leq' t$. For a set S of tree patterns t is the *least generalization* (least common anti-instance) of S, denoted by $lca(S)$, if (1) $q \leq' p$ for any $q \in S$ and for any $p' <' p$, condition (1) does not holds. For a finite set S, $lca(S)$ can be computed in polynomial time [Plo70, LMM88]. A 2-mmg is a natural extension of the notion of least generalization.

Definition 1 Let S be a set of tree patterns. A pair $\{p_0, p_1\}$ is a 2-*minimal multiple generalization*, 2-mmg for short, of S if $L(\{p_0, p_1\})$ is a minimal language containing S within the class $\mathcal{TPL}^2$.

2.2 Logic programs

Let Σ, X and Π be mutually disjoint sets. We refer to Σ as an *alphabet* and to each element of it as a *functor*, to each element of X as a *variable* and to each element of Π as a *predicate symbols*. Each element of Σ and Π has an *arity*. *Terms, atomic formulas (atoms), well-formed formulas, first order language, clauses* and *substitutions* are defined in a usual manner [Llo87]. A *program* is a finite set P of clauses of the form $A\!:\!-A_1, \ldots, A_k$, where $k \geq 0$ and $A, A_1, \ldots, A_k$ are atoms. An arrow "$:-$" and a comma "," are interpreted in a usual way.

We describe the least Herbrand model semantics of logic programs according to Lloyd [Llo87]. Let L be the first order language of a program P. We use the set of all ground atoms of L as the base of interpretations, called the *Herbrand base* $B(L)$. A subset I of $B(L)$ is called *Herbrand interpretation* in the sense that $A \in I$ means A is true in I and $A \notin I$ means A is false in I for an atom $A \in B(L)$. Then, we define the *least Herbrand model* $M(P)$ of P as

$$M(P) = \bigcap \{\, M \subseteq B(L) \mid M \text{ is an Herbrand model of } P\}.$$

This model is also characterized as the set of logical consequence restricted to $B(L)$ of P. In a later section, we will introduce another characterization of the least Herbrand model of a program.

For any program P, there is the unique most specific instance P^* of P where each instance C^* of $C \in P$ also covers the set of true atoms covered by C. For a clause $C = A\!:\!-B_1, \ldots, B_n$ ($n \geq 0$) and $I \subseteq B(L)$, we define the set of covered atom by C, denoted by $C(I)$, as $C(I) = \{\, A \in B(L) \mid A\!:\!-B_1, \ldots, B_n \leq' C \text{ and } B_i \in I \text{ for any } i\}$.

Definition 2 For a program P and $C \in P$, an instance C' of C is a *more specific version of C with respect to P* if $C(M(P)) = C'(M(P))$. A clause C^* is the *most specific version (MSV) of C with respect to P* if C is a most specific (least in the order $\leq'$) in more specific versions of C with respect to P. The *most specific version* (MSV) of P is the program P^* defined as

$$P^* = \{\, C^* \mid C^* \text{ is the MSV of } C \in P \text{ w.r.t. P}\}.$$

Our definition of MSV is slightly different from the original definition proposed by Marriott et. al. [MNL88]. However, they coincide on our class CFTFB_{uniq}.

2.3 Inductive inference from positive data

We fix a finite alphabet Σ. An *indexed family of recursive languages* is a class of nonempty languages $\mathcal{C} = \{L_1, L_2, L_3, \ldots\}$ 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 $\mathcal{C}$. We say that M *identifies $L_i \in \mathcal{C}$ 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 $\mathcal{C}$ is said to be *identifiable in the limit from positive data* if there is an inference machine M such that for any $L_i \in \mathcal{C}$, M identifies L_i in the limit from positive data.

A program P is a *linear Prolog* if for every clause C and each atom B in the body, (1) the number of occurrences of a variable x in the head A is more than or equal to that of x in the B; (2) the size of A is more than or equal to that of B.

Theorem 4 (Shinohara [Shi90]) *For every fixed m, the class of least Herbrand models of linear Prolog with at most m clauses has finite elasticity.*

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 $S_n \subseteq L_{g_{n-1}}$.

$\mathcal{C}$ 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 $\mathcal{C}$ 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 $L_g \in \mathcal{C}$ 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 $|w_1|+\cdots+|w_n|$. In this paper, we simply say *consistent and conservative polynomial time identification in the limit from positive data.*

3 Least Herbrand models of CFTFB$_{uniq}$

In this section, we introduce a subclass of context-free term transformations with at most two clauses for which there is exactly one 2-mmg of $M(P)$ for any program P. We fix a predicate symbol p of arity $m \geq 0$. A Prolog program P defining the predicate p with at most two clauses C_0 and C_1

$$\begin{array}{rcl} C_0 & = & p(s_1,\ldots,s_m), \\ C_1 & = & p(t_1,\ldots,t_m){:-}p(x_1,\ldots,x_m) \end{array}$$

is *a context-free term transformation with a flat base case* (CFTFB) if P satisfies each of (a)–(c):

(a) Every argument s_i $(1 \leq i \leq m)$ of the head of C_0 is either a function symbol of arity 0 or a variable symbol.

(b) All arguments $x_1,\ldots,x_m$ of the body of C_1 are mutually distinct variables.

(c) For every $1 \leq i \leq m$, every argument x_i of the body of C_1 occurs exactly once in the term t_i of the head. Moreover, x_i does not occurs in any arguments t_j $(i \neq j)$ of the head.

We denote by $h(P)$ the set $\{H_0, H_1\}$ of heads of clauses in a CFTFB $P = \{H_0,\ H_1{:-}D\}$.

```
app([],Y,Y).
app([A|X],Y,[A|Z]):- app(X,Y,Z).

suffix(X,X).
suffix(X,[A|Y]):- suffix(X,Y).

plus(X,0,X).
plus(X,s(Y),s(Z)):- plus(X,Y,Z).

lesseq(0,X).
lesseq(s(X),s(Y)):- lesseq(X,Y).
```

Figure 1: Programs in CFTFB$_{uniq}$

Definition 3 For a CFTFB P, if there is at most one 2-mmg of $M(P)$, then we say that P is a *CFTFB$_{uniq}$*.

After some preliminary lemmas, we will give an efficient condition for CFTFB$_{uniq}$, later. We denote by $\mathcal{CFTM}$ the class of least Herbrand models defined by programs in CFTFB$_{uniq}$. This class $\mathcal{CFTM}$ contains infinitely many tree languages over $\{p\}\cup\Sigma$. For instance, the program defining concatenation relation *app*/3 over lists is contained in the class. Moreover, the class CFTFB$_{uniq}$ contains small Prolog programs defining predicates such as addition $plus/3$ over natural numbers represented by the successor function and suffix relation $suffix/2$ over lists (Figure 1). Since CFTFB$_{uniq}$ is a subclass of linear Prolog, the next lemma is immediately derived from Theorem 4.

Lemma 5 $\mathcal{CFTM}$ has finite elasticity.

The membership decision of a string is decidable in polynomial time.

Lemma 6 There is an algorithm that, given an ground atom w and a CFTFB P, decides whether $w \in M(P)$ in polynomial time.

Proof. Since P has at most one clause whose body is not empty, we can decide $w \in M(P)$ by constructing SLD-derivations [Llo87] from $P\cup\{\leftarrow w\}$ where the total number of reduction is at most $2|w|$. □

Proposition 7 Let $P = \{C_0, C_1\}$ be the MSV of a CFTFB such that $C_0 \equiv' H_0$, $C_1 \equiv' H_1{:-}D$. Then,

1. $H_1 \equiv' lca(C_1(M(P)))$, and
2. $D \equiv' lca\{H_0, H_1\}$.

Proof. (2) Prop. 4.1. of [MNL88] says that $D \equiv' lca(M(P))$ for any MSV of logic programs. On the other hand, clearly, $M(P) = C_0(M(P)) \cup C_1(M(P))$. By applying Lemma 4 in Lassez et. al. [LMM88], which says that $lca\{lca(S_1), lca(S_2)\} \equiv' lca(S_1 \cup S_2)$, to this, we obtain that $lca(M(P)) \equiv'$

$lca\{lca(C_0(M(P))), C_1(lca(M(P)))\}$. Since $lcaC_0(M(P))$ is equivalent to the H_0 and $lcaC_1(M(P))$ is equivalent to H_1, we have $lca(M(P)) \equiv' lca\{H_0, H_1\}$ and it is also equivalent to D. □

For a Prolog program P, there exists the unique MSV of P [MNL88]. But, the converse is not true in general.

Lemma 8 There is a one-to-one mapping ϕ from MSVs of CFTFBs to CFTFBs such that for an MSV P^* of CFTFB P, $\phi(P^*) \equiv' P$. Furthermore, this mapping can be computed in polynomial time in the size of P^*.

Proof. Let $P^* = \{H_0,\ H_1{:}-D\}$ be the MSV of a CFTFB. For H_0 is a flat base and all symbols label the root of arguments of H_1 are of arity more than 0, all arguments of $D \equiv' lca\{H_0, H_1\}$ are variables or function symbols of arity 0. Let $H = p(t_1, \ldots, t_m)$ and $D = p(d_1, \ldots, d_m)$. If d_i $(1 \le i \le m)$ is a function symbol of arity 0, then the t_i must be the same symbol d_i; Thus, we replace both occurrences of t_i, d_i by a new variable y_i. If $d_i = d_j$ $(i \neq j)$ are the same variable, each d_k must occur exactly once in the k-th argument t_k $(k = i, j)$. Then, we replace both occurrences of t_k, d_k in k-th arguments of atoms in C_1 by the same new variable y_k for each $k = i, j$, where $y_i \neq y_j$. Let $\phi(P^*)$ be the program obtained by applying this operation until they are not applicable; a CFTFB $\phi(P^*)$ is determined uniquely and it can be computed in polynomial time in P^*. □

For example, given the MSV P^* of a CFTFB

```
p(X,X,a,a),
p(s(Y),s(Y),a,s(W)):- p(Y,Y,a,W),
```

we have the CFTFB $\phi(P^*)$ whose MSV is P^*

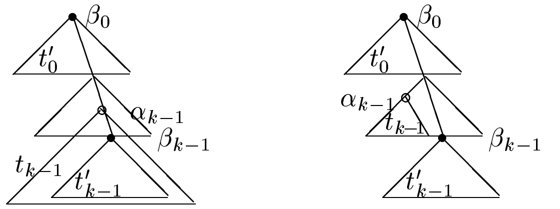
```
p(X,X,a,a),
p(s(U),s(V),Z,s(W)):- p(U,V,Z,W).
```

For a CFTFB $P = \{H_0,\ H_1{:}-D\}$, the tree language defined by P is its least Herbrand model $M(P)$. We characterize this $M(P)$ by an infinite sequence of atoms possibly containing variables; that better reflects the structure of P than the layers generated by the mapping T_P frequently used in logic programming. An infinite sequence $\mathcal{J}_P$ of atoms

$$j_0, j_1, \ldots, j_n, \ldots \ (n \ge 0)$$

is defined as a sequence satisfies the following conditions: (1) $j_0 = H_0$; (2) Let θ be the most general unifier of B and j_{n-1} for a variant $A{:}-B$ of the $H_1{:}-D$; Then, $j_n = A\theta$; (3) There is no variable occurs in both of $j(m)$ and $j(k)$ for any $m \neq k$.

Given P, the sequence $\mathcal{J}_P : j_0, j_1, \ldots$ is uniquely determined up to variable renaming. For instance, the program app/3 in Fig 1 has the sequence $\mathcal{J}_P$



Case(a):$\alpha_{k-1} \sqsubset \beta_{k-1}$ Case(b):$\alpha_{k-1} \not\sqsubseteq \beta_{k-1}$

Figure 2: Two cases in the proof of Lemma 10

$$app([], x_0, x_0),\ app([x_1], y_1, [x_1|y_1]),$$
$$app([x_2, y_2], z_2, [x_2, y_2|z_2]),$$
$$app([x_3, y_3, z_3], w_3, [x_3, y_3, z_3|w_3]), \ldots .$$

Proposition 9 Let P be a CFTFB$_{uniq}$ and $\mathcal{J}_P$: $j_0, j_1, \ldots$. Assume that an alphabet Σ has more than two symbols. Then, P has the following properties:

(a) $M(P)$ is equivalent to the infinite union defined by $\mathcal{J}_P$, that is, $M(P) = \bigcup_{n \ge 0} L(j_n)$.

(b) If a union $L(p) \cup L(q)$ contains $M(P)$, then, either $L(j_n) \subseteq L(p)$ or $L(j_n) \subseteq L(q)$ holds for each $n \ge 0$.

Proof. (a) By Lemma 8 in Lassez et. al. [LMM88], $lca(L(j_n)) \equiv' j_n$ for any $n \ge 0$. Thus, our semantics coincides to the standard fixed point one defined by T_P mapping [Llo87]. (b) $M(P) \subseteq L(p) \cup L(q)$ implies $L(j_n) \subseteq L(p) \cup L(q)$ for each $n \ge 0$. Since $\sharp\Sigma > 2$, we get the result by the compactness w.r.t.containment in Proposition 2. □

By Proposition 9, we have the following lemmas, which will be used for proving that the class CFTFB$_{uniq}$ is polynomial time decidable and that 2-mmg of $M(P)$ coincides to heads for CFTFB$_{uniq}$ P.

We first show a technical lemma on trees with a periodic structure like a linear list. For an atom $A = p(t_1, \ldots, t_m)$ $(m \ge 0)$ and an index $1 \le i \le m$, we denote by A/i the i-th argument t_i of A.

Lemma 10 Let $P = \{H_0,\ H_1{:}-D\}$ be CFTFB and $\mathcal{J}_P : j_0, j_1, \ldots$. Assume that $H_0/i = H_0/j$ for some $i \neq j, 1 \le i, j \le m$ and $n \ge 2$. Then, $j(n)/i = j(n)/j$ implies $H_1/i = H_1/j$.

Proof. We show that if $j(n)/i = j(n)/j$, then $\alpha = \beta$, where α is the "occurrence" (position or node) of the variable D/i in H_1/i and β is that of D/j in H_1/j. We can show this considering only the shapes of $j(n)/i, j(n)/j$ not labels of nodes (Figure 2). However, we omit the detailed proof. □

If $n = 1$, this lemma does not holds. We show an example: for the program

```
p(a,a),
p(f(a,X),f(Y,a)):- p(X,Y),
```

$H_0/1 = H_0/2$ and $j(1)/1 = j(1)/2$, but $H_1/1 \neq H_1/2$. This parameter 2 in the formula $n \geq 2$ dominates the number of 2-mmg's of $M(P)$

Lemma 11 Let P be a CFTFB, $\mathcal{J}_P : j_0, j_1, \ldots$ and $\sharp\Sigma > 2$. Then, there are at most two 2-mmg's of $M(P)$. Moreover, they are $\{lca\{j_0\}, lca(\bigcup_{n\geq 1}\{j_n\})\}$ and $\{lca\{j_0, j_1\}, lca(\bigcup_{n\geq 2}\{j_n\})\}$.

Proof. From the construction of $\mathcal{J}_P : j_0, j_1, \ldots$, we have Claim.

Claim . For any $k \geq 2$, $lca(\mathcal{J}_P) \leq' lca\{j_0, j_k\}$. Moreover, if heads of P are unifiable, then for any $k \geq 1$, $lca(\mathcal{J}_P) \leq' lca\{j_0, j_k\}$.

Proof. It is derived from Lemma 10 and fact that all arguments of $lca\{H_0, H_1\}$.

Assume that $M(P) \subseteq L(\{q_0, q_1\})$ for some $\{q_0, q_1\}$. Here, we denote $s \leq' t$ for any $s \in S$ by $S \leq' tp$. By Proposition 9 (b), each $L(j_n)$ must be included in one of $L(q_0)$ and $L(q_1)$. Assume that $j_0 \leq' q_0$ without loss of generality. Thus, there are four cases:

(1) $\{j_0, j_1, \ldots, j_n, \ldots\} \leq' q_0$,

(2) $\{j_0, j_k\} \leq' q_0$ for some $k \geq 2$,

(3) $\{j_0, j_1\} \leq' q_0$ and $lca(\bigcup_{n\geq 2}\{j_n\}) \leq q_1$,

(4) $\{j_0\} \leq q_0$ and $lca(\bigcup_{n\geq 1}\{j_n\}) \leq q_1$,

In the (1),(2) and (4), $L(\{j_0, lca(\bigcup_{n\geq 1}\{j_n\})\}) \subseteq L(\{q_0, q_1\})$ by Proposition 8 and Claim 1. Since languages and their representation correspond for $\mathcal{TPL}^2$, the result immediately follows. □

The class CFTFB$_{uniq}$ is polynomial time decidable.

Theorem 12 *We can decide, given a program P, whether P is CFTFB$_{uniq}$ in polynomial time of $|P|$, where $|P|$ is the size of P as an expression.*

Proof. Since we can easily check whether P is CFTFB in polynomial time in $|P|$, we assume that P is a CFTFB without loss of generality. Let h_0, h_1 be heads of P. Using Lemma 7 and Claim in the proof of Lemma 7, we can prove the following claim.

Claim . Let P be a CFTFB and $\mathcal{J}_P : j_0, j_1, \ldots$. Then, P is a CFTFB$_{uniq}$ iff $lca\{j_1, j_2\} \leq' lca\{j_0, j_1\}$.

Proof. It is clear from Lemma 11.

We can check the condition in polynomial time by using the term matching algorithm and the least generalization algorithm. Hence, the result follows from the above claim. □

```
procedure M;
    input: an infinite sequence w_1, w_2, ...of words;
    output: an infinite sequence g_1, g_2, ...of programs;
begin
    Set S to be the empty sample;
    Set H to be the empty program;
    repeat
        read next example w and add it to S;
        if w ∉ M(H) then
        begin
            let {h_0, h_1} be a 2-mmg(S);
            find a hypothesis P* consistent with S
                whose heads are {h_0, h_1};
            if found then let H be φ(P*);
            if not found then let H be {h_0, h_1};
            output H;
        end;
    forever; /* main-loop*/
end.
```

Figure 3: Algorithm for inferring CFTFB$_{uniq}$

4 Inferring Heads

In this section, we present an algorithm for inferring a subclass of context-free transformations by using the subprocedure 2-mmg.

We show our inference algorithm in Fig 3. If the inference algorithm M receives the current sample S, it first tries to identify the pair of heads of the unknown program by computing a 2-mmg of S. Once M has a 2-mmg $\{h_0, h_1\}$ of S, it searches possible hypotheses, which is the MSV of a CFTFB$_{uniq}$, consistent with the sample S by adding a body to one of the pair. If M succeeds to construct a hypothesis program in the hypothesis space, then it outputs the program. If M fails to add a body, it outputs 2-mmg $\{h_0, h_1\}$ itself as an approximation of $M(P)$.

The key to an efficient inference algorithm is to identify heads of the unknown program P in polynomial time.

Proposition 13 (Arimura et. al. [ASO91]) Assume that the alphabet Σ has more than two symbols. Then, there is an algorithm that, given a sample S, computes one of 2-mmg of S in polynomial time in the total size of S.

For the completeness of our inference machine, we assume that any output $\{p_0, p_1\}$ is reduced; In fact, the 2-mmg algorithm in [ASO91] satisfies this requirement.

Theorem 14 *Let P be a CFTFB$_{uniq}$ and $\sharp\Sigma > 2$. Assume P^* be the MSV of P. Then, the set $h(P^*)$ is the unique 2-mmg of $M(P)$.*

Proof. Since a CFTFB$_{uniq}$ has the unique 2-mmg of $M(P)$, two pairs in Lemma 11 represent the same union.

By compactness, these representation coincide, not only languages defined by them.Thus, it immediately follows from Lemma 7 that the pair $\{lca\{j_0\}, lca(\bigcup_{n\geq 1}\{j_n\})\}$ is the head of MSV. □

The inference algorithm finds a 2-mmg pair as a candidate of heads, it searches a context-free transformation with the given pair as heads which is consistent with examples by adding a goal to the body of the pair and outputs the for program as a conjecture.

The following two lemmas allow us that we can reduce the containment relation $\subseteq$ on $\mathcal{CFTM}$ to that on $\mathcal{TPL}^2$.

Lemma 15 Let P_* and P'_* be MSVs of some CFTFB_{uniq} with $h(P_*) = h(P'_*)$. Then, $M(P_*) \not\subset M(P'_*)$.

Proof. For CFTFB_{uniq}, a similar result can be shown.

Claim1. Let P and P' be CFTFB_{uniq} with $h(P) = h(P')$. Then, $M(P) \not\subset M(P')$.

By Lemma 8, we have the following claim.

Claim2. If P_* and P'_* be CFTFB_{uniq} with $h(P_*) = h(P'_*)$, then $\phi(P)$ and $\phi(P')$ be CFTFB_{uniq} with $h(\phi(P)) = h(\phi(P'))$.

Then, the result is derived from Claim 1 and Claim 2. □

Lemma 16 Let P_* be the MSV of a CFTFB_{uniq} and $\{h_0, h_1\}$ be a pair such that $\{h_0, h_1\} \equiv' h(P_*)$. If $M(P_*) \subseteq L(\{h_0, h_1\})$, then $L(h(P_*)) \subset L(\{h_0, h_1\})$.

Proof. By Lemma 11, if $M(P_*) \subseteq L(\{h_0, h_1\})$, then $L(h(P_*)) \subseteq L(\{h_0, h_1\})$. By compactness with respect to containment, $L(h(P_*)) \subseteq L(\{h_0, h_1\})$ only if $h(P_*) \equiv' \{h_0, h_1\}$. Thus, the result follows from the assumption. □

Lemma 17 Let S be a set of ground atoms and $\{h_0, h_1\}$ be any 2-mmg of S. If there is the MSV P_* of a CFTFB_{uniq} such that $S \subseteq M(P_*)$ and $\{h_0, h_1\} \equiv' h(P_*)$, then $M(P_*)$ is a minimal language containing S within $\mathcal{CFTM} \cup \mathcal{TPL}^2$.

Proof. Assume that $M(P_*)$ is not minimal within $\mathcal{TPL}^2$, that is, there exists some $L(\{q_0, q_1\}) \in \mathcal{TPL}^2$ such that $S \subseteq L(\{q_0, q_1\}) \subset M(P_*)$. Since the union $L(\{h_0, h_1\})$ defined by heads contains $M(P_*)$, this shows that $L(\{h_0, h_1\})$ is not a 2-mmg of S. This contradicts our assumption for $L(\{h_0, h_1\})$. On the other hand, assume that there exists some P'_* in the MSV of a CFTFB_{uniq} that violates the minimality of $M(P_*)$, that is, $S \subseteq M(P'_*) \subset M(P_*)$. If pairs of heads of P'_*, P_* are pairwise distinct modulo renaming of variables, Claim 1 shows that $\{h'_0, h'_1\}$ violates the minimality of $\{h_0, h_1\}$, which is contradiction. Otherwise, $h'_0 \equiv' h_0$ and $h'_1 \equiv' h_1$ since both of P'_* and P_* are MSV of some CFTFB. Thus, we get $M(P'_*) \subset M(P_*)$ applying Theorem 15. However, it is impossible. Hence, $M(P_*)$ is minimal with in both of $\mathcal{TPL}^2$ and $\mathcal{CFTM}$. □

If the inference algorithm can not find any consistent hypothesis with examples whose heads are the given pair, then it simply output the pair of atoms as an approximation of the model by unions of two tree pattern languages.

Lemma 18 Let S be a set of ground atoms and $\{h_0, h_1\}$ be any 2-mmg of S. If there is no MSV P_* of a CFTFB_{uniq} such that $S \subseteq M(P_*)$ and $\{h_0, h_1\} \equiv' h(P_*)$, then $L(\{h_0, h_1\})$ is a minimal language containing S within $\mathcal{CFTM} \cup \mathcal{TPL}^2$.

Proof. Assume that $L(\{h_0, h_1\})$ is not minimal within $\mathcal{TPL}^2$. It is impossible because $L(\{h_0, h_1\})$ is 2-mmg of S. Therefore, we assume that there exists some P'_* in MSV of CFTFB_{uniq} that violates the minimality of $L(\{h_0, h_1\})$. Thus, $S \subseteq M(P'_*) \subset L(\{h_0, h_1\})$. If $h(P'_*) \not\equiv' L(\{h_0, h_1\})$, by a similar argument in Claim 2, $h(P'_*)$ violates the minimality of $\{h_0, h_1\}$. Thus, it is impossible. If $h(P'_*) \equiv' L(\{h_0, h_1\})$, this contradict the assumption that there is no MSV of CFTFB_{uniq} program consistent with S whose heads are $\{h_0, h_1\}$. Hence, the pair $\{h_0, h_1\}$ is also minimal within both of $\mathcal{TPL}^2$ and $\mathcal{CFTM}$ containing S. □

Theorem 19 *Assume that Σ has more than two symbols. For each iteration of main loop in the inference algorithm M, it outputs a program that represents a minimal language containing S within the class $\mathcal{CFTM}_{uniq} \cup \mathcal{TPL}^2$.*

Proof. When $h_0 \equiv' h_1$, the algorithm outputs $\{h_0, h_1\}$ and the correctness will easily be shown by using Lemma 16. The output of the inference algorithm is the MSV of a CFT-FB program P only in case of Lemma 17 and that is a pair $\{h_0, h_1\}$ only in case of Lemma 18 and in case of $h_0 \equiv' h_1$. Hence, the theorem is proved. □

From a discussion in section 2, this theorem ensures that the algorithm works consistently and conservatively.

5 Finding A Consistent Hypothesis

Theorem 20 *Let p be a predicate symbol of fixed arity $m \geq 0$ and Σ has more than two symbols. Then, there is an algorithm that, given a pair $\{h_0, h_1\}$ and a sample S, computes the MSV of a CFTFB_{uniq} program such that $S \subseteq M(P^*)$ and $\{h_0, h_1\} \equiv' h(P^*)$ in polynomial time with respect to $|S|$, where $|S|$ is the total size of examples in S.*

Proof. Assume that $h_0 \not\leq' h_1$ and $h_1 \not\leq' h_0 s$ for $\{h_0, h_1\}$ because we assumed our 2-mmg algorithm outputs only reduced pairs. If there is the MSV P^*

$$\begin{array}{rcl} C_0 & \equiv' & p(s_1, \ldots, s_m), \\ C_1 & \equiv' & p(t_1, \ldots, t_m){:-}p(r_1, \ldots, r_m) \end{array}$$

of a Prolog program with two clauses such that

(1) $h(P^*) \equiv' \{h_0, h_1\}$,

(2) P^* is the MSV of a CFTFB_{uniq}, and

(3) $S \subseteq M(P^*)$.

then we can additionally assume the following conditions by Proposition 7 and an assumption for outputs of the 2-mmg algorithm.

(4) The head $p(t_1, \ldots, t_m)$ has at least one functor of arity 0 as an argument and $p(s_1, \ldots, s_m)$ does not.

(5) $p(r_1, \ldots, r_m) \equiv' lca\{h_0, h_1\}$, and r_i is a subterm of t_i for each $1 \leq i \leq m$.

Let $\mathcal{H}(\{h_0, h_1\}, S)$ be the set of programs P^* which satisfies (1),(4) and (5). Then,

Claim 1. $\sharp(\mathcal{H}(\{h_0, h_1\}, S))$ is bounded by $O(n^c)$ for some constant $c \geq 0$, where n is the total size of examples in S.

Proof. Assume that P^* is a program in $\mathcal{H}(\{h_0, h_1\}, S)$. Let $C_1 = H_1{:-}D$. Since $S \cap C_1(M(P)) \neq \emptyset$, we can assume that $|H_1| \leq |S|$ by condition (3). By (5), each r_i is a subterm in t_i. Thus, the number of different choices of its occurrences is at most $|t_i|$. Thus, $\sharp(\mathcal{H}(\{H_0, h_1\}, S)) \leq |h_1|^m$.

The algorithm computes the MSV of a CFTFB_{uniq} program P^* such that $S \subseteq M(P^*)$ and $\{h_0, h_1\} \equiv' h(P^*)$ in the following way. The algorithm first enumerates instances P of CFT-FB in $\mathcal{H}(\{h_0, h_1\}, S)$. Then it checks whether P^* satisfies both of (2) $\phi(P^*)$ is a CFTFB_{uniq} and (3) $S \subseteq M(P^*)$; Checking (2) can be done in polynomial time in $|P^*|$ by checking whether $\phi(P^*) \in \mathrm{CFTFB}_{uniq}$ described in Theorem 12; and Checking (3) can be done in polynomial time in $|S|$ and $|P|$ by using membership decision procedure in Lemma 6. If such a CFT-FB P^* is found, the algorithm returns P^* □

Corollary 21 *The algorithm M updates a hypothesis in polynomial time in the total size of current sample.*

Proof. It is immediately derived from Lemma 6, Lemma 8, Proposition 13 and Theorem 20. □

Since $\mathcal{CFTM}_{uniq} \cup \mathcal{TPL}^2$ is a subclass of linear Prolog [Shi90], it has finite elasticity [Wri89a] as shown in Lemma 5. By a similar argument in Angluin [Ang79], an inference machine always outputs a minimal language containing the current sample identifies the class $\mathcal{C}$ with finite elasticity from positive data consistently and conservatively [ASO91]. Hence, we show the main result of this paper from Theorem 19 and Corollary 21.

Theorem 22 *Let p be a predicate symbol of fixed arity $m \geq 0$ and Σ has more than two symbols. Then, the class $\mathcal{CFTM} \cup \mathcal{TPL}^2$ is consistently and conservatively polynomial time identifiable in the limit from positive data.*

6 Conclusion

We showed that the class of tree languages defined by CFTFB_{uniq} together with TP^2 is consistently and conservatively polynomial time identifiable in the limit from positive data under some restriction on the size of an alphabet.

Our algorithm uses the 2-mmg algorithm (2-mmg) to infer a pair of heads of the unknown program efficiently. This technique can be considered as an extension of the method proposed in [Ish88]. Where, to infer heads of clauses, after dividing examples into several subsets, an inference algorithm generalize each subsets by the least generalization (lg) algorithm [Plo70]; However, since the number of possible partitions of examples may be exponential, this algorithm can not achieve efficiency. Hence, since 2-mmg can be considered as a natural extension of lg, it seems to be useful employing 2-mmg in inductive inference systems [Ish88, Mug88] which use lg.

In this paper, we considered only inference machines working consistently and conservatively with polynomial time updating hypothesis and did not consider other inference criterions. Our algorithm efficiently updates a hypothesis, but the exponentially many total updates may be required. There is another good criterion for efficient identification in the limit (Pitt [Pit89]); 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. Since our class considered in this paper shares some properties with pattern languages [Ang79, ASO91], unfortunately, the class $\mathcal{CFTM} \cup \mathcal{TPL}^2$ seems not to be identifiable in the limit from positive data under Pitt's criterion.

Acknowledgements

The first author would like to thank Setsuo Arikawa for his encouragement, Setsuko Otsuki and Akira Takeuchi for supporting the work on this paper.

References

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

[Ang82] D. Angluin. Inference of reversible languages. *Journal of the ACM*, Vol. 29, pp. 741–765, 1982.

[ASO91] H. Arimura, T. Shinohara, and S. Otsuki. Polynomial time inference of unions of tree pattern languages. In *Proceedings of the Second Workshop on Algorithmic Learning Theory*, pp. 105–114, 1991.

[Ish88] H. Ishizaka. Model inference incorporating generalization. *Journal of Information Processing*, Vol. 11, No. 3, pp. 206–211, 1988.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming*. Springer Verlag, second edition, 1987.

[LMM88] 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.

[MNL88] K. Marriott, L. Naish, and J-L. Lassez. Most specific logic programs. In *Proc. 5th ICLP*, pp. 909–923. MIT Press, 1988.

[Mug88] S. Muggleton. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, pp. 339–352, 1988.

[Pit89] L. Pitt. Inductive inference, dfas, and computational complexity. In K. P. Jantke, editor, *Proceedings of International Workshop on Analogical and Inductive Inference*, 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, pp. 153–163. Edinburgh University Press, 1970.

[Sak89] Y. Sakakibara. An efficient learning of context-free grammars from positive structural examples. Technical report, FUJITSU LABORATORIES, IIAS, 1989. To appear in Information and Computation.

[Sha81] E.Y. Shapiro. Inductive inference of theories from facts. Technical Report 192, Yale University, Department of Computer Science, 1981.

[Shi90] T. Shinohara. Inductive inference of monotonic formal systems from positive data. In *Proceedings of the First International Workshop on Algorithmic Learning Theory*, pp. 339–351, 1990.

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

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