ICOT Technical Report: TR-330

TR-330

Inferring Parsers of Context-Free Languages from Structural Examples

by

Y. Sakakibara

November, 1987

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Inferring Parsers of Context-Free Languages from Structural Examples

by

Yasubumi SAKAKIBARA*

* Research Associate, Fundamental Informatics Section, International Institute for Advanced Study of Social Information Science, FUJITSU LIMITED
140 Miyamoto, Numazu, Shizuoka 410-03, JAPAN

Abstract

We consider a grammatical inference of context-free languages from their structural descriptions. In the context of inferring *parsers*, the *structure* of the grammar inferred is significant. The structure of a context-free grammar is described by the *shapes* of derivation trees, called *skeletons* which are derivation trees from the grammar with non-labeled nodes. It is known that the set of derivation trees of a context-free grammar is rational, and the set of skeletons of a context-free grammar is also rational. Based on this fact, by extending an efficient inductive inference algorithm for finite automata to the one for tree automata, we can get an efficient inductive inference algorithm for parsers of context-free languages. A grammar (or parser) inferred by the algorithm is not only a *correct* grammar which correctly generates the language but also assign a *correct* structure on the sentences of the language.

1. Introduction

In this paper, we will study the inductive inference of parsers (or grammars) of context-free languages from examples of their structural descriptions. Inductive inference of formal languages is formally defined by Gold [8]. Especially, the problem of identifying a "correct" grammar for a language from finite examples of the language is known as the grammatical inference problem. In the context of grammatical inference, a "correct" grammar only means a grammar which correctly generates the language. However when we consider the problem of identifying a parser for a language, the *structure* of the grammar identified is more significant. Consider the following example from [12]. The grammar G_1 below, which specifies the set of all valid arithmetic expressions involving a variable "v" and the operations of multiplication "×" and addition "+".

$$S_1 \rightarrow v$$
$$S_1 \rightarrow A_1 v$$
$$A_1 \rightarrow v+ \qquad\qquad \text{(the grammar } G_1\text{)}$$
$$A_1 \rightarrow v\times$$
$$A_1 \rightarrow v+A_1$$
$$A_1 \rightarrow v\times A_1$$

For example, the parse tree for $v\times v+v\times v$ is shown in Figure 1.1. However the structure assigned by grammar G_1 to this sentence is semantically meaningless. The same language can be described by grammar G_2 below in a meaningful manner.

$$S_2 \rightarrow E_2$$
$$E_2 \rightarrow F_2$$
$$E_2 \rightarrow F_2+E_2 \qquad\qquad \text{(the grammar } G_2\text{)}$$
$$F_2 \rightarrow v$$
$$F_2 \rightarrow v\times F_2$$

The structure assigned by G_2 to the same sentence is shown in Figure 1.2. Here the phrases are all significant in terms of the rules of arithmetic.

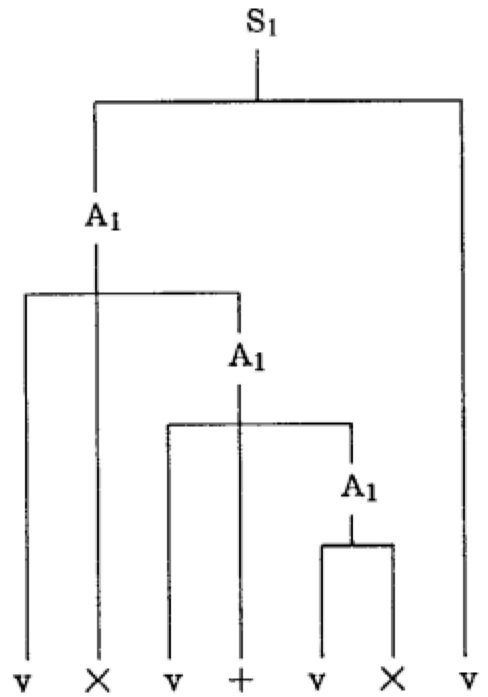

Figure 1.1

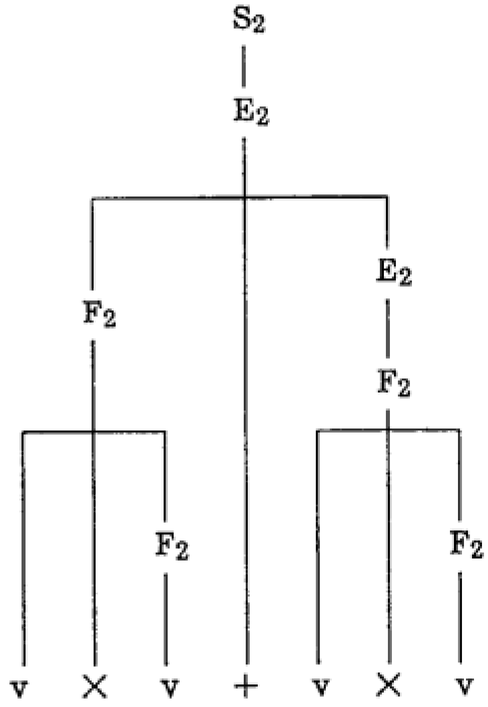

Figure 1.2

Although G_1 and G_2 are weakly equivalent, this fact is not very relevant from a practical point of view since it would be unusual to consider such a grammar as G_1 which structures sentences in a nonsignificant manner.

Thus in the context of inferring a parser, since a grammar inferred is intended for use in a practical situation entailing the translation or interpretation of sentences in a compiler, it is necessary that a grammar inferred must not only generate the unknown language, but also assign a meaningful structure on the sentences of the language. To do so, it is necessary for us to assume that information on the structure of the language is available to the inference algorithm. In the case of context-free languages, the structure of the languages is usually described by the *shapes* of the derivation trees. Such structural descriptions are called *skeletons*. A skeleton is a kind of tree whose interior nodes have no label.

On the other hand, the set of derivation trees of a context-free grammar is rational, where a *rational* set of trees is a set of trees which can be recognized by some tree automaton. Furthermore, the set of skeletons of a context-free grammar is also rational. Based on this fact, the problem of inductive inference of parsers of context-free languages from the sentences and structures is reduced to the problem of inductive inference of tree automata. Then by extending an inductive inference algorithm for finite automata [1] to the one for tree automata, we can get an efficient inductive inference algorithm for parsers of context-free languages. A grammar (or parser) inferred by the algorithm is not only a *correct* grammar which correctly generates the language but also assign a *correct* structure on the sentences of the language. Furthermore, the time complexity of the algorithm is a polynomial order with respect to the size of input examples.

2. Basic definitions of tree

Definition We denote N the set of positive integers. Dom is a *tree domain* iff it satisfies

a) $\mathrm{Dom} \subseteq N^*$ and Dom is finite,

b) Dom is prefix-closed, i.e. if $m, n \in N^*$ and $mn \in \mathrm{Dom}$ then $m \in \mathrm{Dom}$,

c) $ni \in \mathrm{Dom}$ implies $nj \in \mathrm{Dom}$ for $1 \leq j \leq i, j \in N$.

A *direct successor* (*direct predecessor*) of a node x is a node y, where $y = xi$ ($yi = x$) for $i \in N$. A *terminal node* in Dom is one which has no direct successor. The *frontier* of Dom, denoted *frontier(Dom)*, is the set of all terminal nodes in Dom. The *interior* of Dom, denoted *interior(Dom)*, is Dom − frontier(Dom).

Definition The *depth* of $n \in \mathrm{Dom}$ is recursively defined as

$\mathrm{depth}(n) = 0 \quad \text{if } n = \varepsilon$

$\mathrm{depth}(ni) = \mathrm{depth}(n) + 1 \quad \text{for } i \in N.$

If t is a tree domain, then $\mathrm{depth}(t) = \max\{\mathrm{depth}(i) : i \in t\}$.

Definition [13] An *alphabet* is a finite nonempty set of symbols. A *ranked alphabet* Γ is a finite set of symbols associated with a relation $r_\Gamma \subseteq \Gamma \times \{0,1,2,...,m\}$. The r_Γ is called the *rank* of Γ. For each $n \geq 0$, the subset $\{a \in \Gamma : (a, n) \in r_\Gamma\}$ is denoted by Γ_n. The rank here is not necessarily a function. In many cases the symbols in Γ_n will be considered as *function symbols*. The rank of a function symbol is called its *arity* and a symbol of arity 0 is called a *constant symbol*.

Definition ([3]) A *tree* over a finite ranked alphabet Γ is a mapping $t : \mathrm{Dom} \to \Gamma$, which labels the nodes of the tree domain Dom. We require the following condition : if $t(m) = f \in \Gamma_n$, then for $i \in N$, $mi \in \mathrm{Dom}(t)$ iff $1 \leq i \leq n$. Γ^T denote the set of all trees over Γ.

If we consider Γ as a set of function symbols, the finite trees over Γ can be identified with *well-formed terms* over Γ and written linearly with commas and parentheses. Within a proof or a theorem, we shall only write down well-formed terms to represent well-formed trees. Hence when declaring "let t be of the form $f(t_1,\ldots,t_n)$ ..." we also declare that f is of arity n and this allows n to be 0 (in this case $(t_1,\ldots,t_n)$ is the empty sequence, i.e. $t = f$).

Definition Let $t = f(t_1,\ldots,t_n)$ be a tree over Γ. The *frontier* of t, denoted *frontier(t)*, is a string over Γ_0 recursively defined as

$$\begin{aligned} \mathrm{frontier}(t) &= f \quad \text{for } n = 0 \text{ and } f \in \Gamma_0 \\ &= \mathrm{frontier}(t_1)\cdots\mathrm{frontier}(t_n) \quad \text{for } n > 0. \end{aligned}$$

Let T be a set of trees. The *frontier set* of T, denoted *Front(T)*, is $\mathrm{Front}(T) = \{\mathrm{frontier}(t) : t \text{ is in } T\}$.

Definition If $t \in \Gamma^T$, then the *subtree* of t at n, where n is in the domain of t ($n \in \mathrm{Dom}(t)$), is defined as $t/n = \{(i, x) : (ni, x) \in t\}$. For $t \in \Gamma^T$ and $n \in \mathrm{Dom}(t)$, the *replacement* at n with a tree u is defined as $t(n \leftarrow u) = \{(m, x) : t(m) = x \text{ and } n \not\leq m\} \cup \{(ni, x) : u(i) = x \text{ and } i \in \mathrm{Dom}(u)\}$. The *replacement (substitution) of terminal nodes* labeled $c \in \Gamma$ with a tree u is defined as $t(c \leftarrow u) = \{(m, x) : t(m) = x \text{ and } x \neq c\} \cup \{(ni, x) : t(n) = c, u(i) = x \text{ and } i \in \mathrm{Dom}(u)\}$.

Definition Let \$ be a new symbol of arity 0 that we add to Γ. $(\Gamma\cup\{\$\})^T$ denotes the set of all trees over $\Gamma\cup\{\$\}$. Especially we are interested in the subset Sub of $(\Gamma\cup\{\$\})^T$ which is the set of all trees $t\in(\Gamma\cup\{\$\})^T$ such that t exactly contains one \$-symbol. We use the notation $\Gamma_{\$}^T$ for the Sub. For trees $t\in\Gamma^T$ and $s\in\Gamma_{\$}^T$, we define an operation "#" to replace the node labeled \$ of s with t by $s\#t=s(\$\leftarrow t)$ (like concatenation of strings).

Definition A *skeletal alphabet* Sk is a ranked alphabet consisting of the singleton $\{\sigma\}$ of the special symbol σ associated with a relation $r_{Sk}\subseteq\{\sigma\}\times\{1,2,\ldots,m\}$. A *skeleton* over an alphabet A is a mapping s : $Dom\rightarrow A\cup Sk$ where σ is not in A, mapping frontier(Dom) to A and interior(Dom) to Sk. Let t be a tree over Γ. The *skeletal* (or *structural) description* of t, denoted *s(t)*, is a skeleton over Γ_0 such that

$$s(x) = t(x) \quad \text{for } x\in\text{frontier(Dom)}$$
$$\quad = \sigma \quad \text{for } x\in\text{interior(Dom)}.$$

We require that if $t(x)=f$ of arity $n\geqq 1$ for $x\in$interior(Dom), then $s(x)=\sigma\in Sk_n$. Let T be a set of trees. The *corresponding skeletal set*, denoted *S(T)*, is $S(T)=\{s(t) : t \text{ is in } T\}$.

Thus a skeleton is a tree defined over $\Gamma_0\cup Sk$ which has a special symbol σ for the interior nodes. The skeletal description of a tree preserves the structure of the tree, but not the label names describing that structure.

3. Tree automaton and context-free grammar

Definition ([14]) A *deterministic (frontier to root) tree automaton* over Γ is a 4-tuple $T_A=(Q, \Gamma, \delta, F)$, where

a) Q is a nonempty finite set of states,

b) Γ is a nonempty finite ranked alphabet,

c) $\delta=(\delta_0,\delta_1,\ldots,\delta_m)$ is a state transition function such that

$$\delta_k : \Gamma_k\times Q^k\rightarrow Q \quad (k=0,1,\ldots,m),$$

d) $F\subseteq Q$ is the set of final states.

If δ is a state transition function from $\Gamma_k \times Q^k$ to 2^Q, then T_A is *nondeterministic*. δ can be extended to Γ^T by letting :

$$\delta(f(t_1,\dots,t_k)) = \delta_k(f, \delta(t_1),\dots,\delta(t_k)) \quad \text{for } k>0 \text{ and } f\in\Gamma_k,$$
$$= \delta_0(f) \quad \text{for } k=0 \text{ and } f\in\Gamma_0.$$

The tree t is *accepted* by T_A iff $\delta(t)\in F$. The set of trees accepted by T_A is the subset $L(T_A)$ of Γ^T defined as : $L(T_A)=\{t : \delta(t)\in F\}$. A subset L of Γ^T is called *rational* iff there exists some automaton T_A such that $L=L(T_A)$.

Definition A *deterministic (frontier to root) tree automaton* over Γ is a 4-tuple $T_A=(Q, \Gamma, \delta, F)$, where

a) Q is a nonempty finite set of states,

b) Γ is a nonempty finite ranked alphabet,

c) $\delta=(\delta_0,\delta_1,\dots,\delta_m)$ is a state transition function such that

$$\delta_k : \Gamma_k \times (Q\cup\Gamma_0)^k \rightarrow Q \quad (k=1,2,\dots,m),$$
$$\delta_0(a) = a \quad \text{for } a\in\Gamma_0,$$

d) $F\subseteq Q$ is the set of final states.

In this definition, the labels on the frontier are taken as "initial" states. Any rational set which includes no constant can be recognized by tree automaton defined by the second definition. In the rest of the paper, we will use the second definition for tree automata (because any rational set of trees in our target class does not include any constant).

Especially in [11], the second type of tree automata which recognizes sets of skeletons is called *skeletal automata*.

Proposition 3.1 ([14]) *Nondeterministic frontier to root tree automata are no more powerful than deterministic frontier to root tree automata.*

Given a rational set L_R, by the above proposition, there always exists the minimum state deterministic tree automaton which accepts L_R.

Proposition 3.2 (the replacement lemma [15]) *Let* $T_A=(Q, \Gamma, \delta, F)$ *be a tree automaton and* s, s', t *be trees over* Γ. *If* $\delta(s)=\delta(s')$, *then* $\delta(t(x\leftarrow s))=\delta(t(x\leftarrow s'))$ *for* $x\in Dom(t)$.

For the definitions of context-free grammars and languages, we use the notations of [10]. Here we state some basic definitions about context-free grammars.

Definition A *context-free grammar* is denoted $G=(N, \Sigma, P, S)$, where N and Σ are alphabets of *nonterminals* and *terminals*, respectively. We assume that N and Σ are disjoint. P is a finite set of productions; each production is of the form $A\rightarrow\alpha$, where A is a nonterminal and α is a string of symbols from $(N\cup\Sigma)^*$. Finally, S is a special nonterminal called the *start symbol*.

Definition Let $G=(N, \Sigma, P, S)$ be a context-free grammar. We define two relations $\Rightarrow$ and $\Rightarrow^*$ between strings in $(N\cup\Sigma)^*$. If $A\rightarrow\beta$ is a production of P and α and γ are any strings in $(N\cup\Sigma)^*$, then $\alpha A\gamma\Rightarrow\alpha\beta\gamma$. We say that $\alpha A\gamma$ *directly derives* $\alpha\beta\gamma$ in grammar G. Suppose that $\alpha_1,\ldots,\alpha_m$ are strings in $(N\cup\Sigma)^*$, $m\geq 1$, and $\alpha_1\Rightarrow\alpha_2$, $\alpha_2\Rightarrow\alpha_3,\ldots,\alpha_{m-1}\Rightarrow\alpha_m$. Then we say $\alpha_1\Rightarrow^*\alpha_m$ or α_1 *derives* α_m in grammar G. That is, $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$. The *language generated* by G, denoted *L(G)*, is $\{w : w \text{ is in } \Sigma^* \text{ and } S\Rightarrow^* w\}$.

A finite set of nonterminals and terminals $N\cup\Sigma$ can be viewed as a ranked alphabet Γ, where $\Sigma=\Gamma_0$. Then

Definition Let $G=(N, \Sigma, P, S)$ be a context-free grammar. For A in $N\cup\Sigma$, the set $D_A(G)$ of trees over $N\cup\Sigma$ is recursively defined as :

$$D_A(G) = \{a\} \quad \text{for } A=a\in\Sigma,$$
$$= \{A(t_1,\ldots,t_k) : A\Rightarrow B_1\cdots B_k,\ t_i\in D_{B_i}(G)\ (1\leq i\leq k)\} \quad \text{for } A\in N.$$

A tree in $D_A(G)$ is called a *derivation tree* of G from A. For the set $D_S(G)$ of derivation trees of G from the start symbol S, the S-subscript will be deleted.

Note that $D_A(G)$ for $A \in N$ is a set of trees with depth at least 1.

Proposition 3.3 ([14]) *Let* L_R *be a rational set of trees. Then* Front(L_R) *is a context-free language.*

Proposition 3.4 ([11]) *Let* $G=(N, \Sigma, P, S)$ *be a context-free grammar. Then* D(G) *is a rational set of trees. Furthermore,* S(D(G)) *is a rational set of trees over* $\Sigma \cup Sk$.

Thus the structural descriptions of a context-free grammar constitute a rational set. It is obvious that L(G) = Front(D(G)) = Front(S(D(G))).

For a context-free grammar G, S(D(G)) corresponds to the structural descriptions, called *skeletal structural descriptions*, of L(G). Then

Definition Two context-free grammars G_1 and G_2 are said to be *equivalent* if $L(G_1)=L(G_2)$ (i.e., they generate the same language). Two context-free grammars G_1 and G_2 are said to be *structurally equivalent* if $S(D(G_1))=S(D(G_2))$.

Another structural descriptions of a context-free language can be described by means of a parenthesis grammar.

Definition Let G be a context-free grammar with a set of productions $P=\{A_i \rightarrow \beta_i : 1 \leq i \leq n\}$. Then [G], the *parenthesized version* of G is the grammar with a set of productions $\{A_i \rightarrow [\beta_i] : 1 \leq i \leq n\}$ where "[" and "]" are special brackets that are not terminal symbols of G. [G] is called a *parenthesis grammar.*

Then the following another definition for structurally equivalence is found in [12].

Definition Two context-free grammars G_1 and G_2 are said to be *structurally equivalent* if $L([G_1])=L([G_2])$.

We can easily verify that those two different definitions for structurally equivalence are equivalent.

Definition $G=(N, \Sigma, P, S)$ is a *wide-sense context-free grammar* if N and Σ are alphabets of nonterminals and terminals respectively, N and Σ are disjoint, P is a finite set of productions, each production is of the form $A \rightarrow \alpha$, where A is a nonterminal and α is a string of symbols from $(N \cup \Sigma)^*$, and $S \subseteq N$ is the set of starting symbols.

In this definition, G is the usual context-free grammar but may have more than one starting symbol.

Proposition 3.5 ([11]) *For each wide-sense context-free grammar* G, *there is a context-free grammar* G' *with a unique start symbol such that* G' *is structurally equivalent to* G.

Now we show two important propositions which connect a context-free grammar with a tree automaton.

Definition-A Let $G=(N, \Sigma, P, S)$ be a wide-sense context-free grammar. The *corresponding (nondeterministic) tree automaton* $T_A(G)=(Q, \Sigma \cup Sk, \delta, F)$ over $\Sigma \cup Sk$ is defined with state set Q, final states F, and state transition function δ as follows.

$Q = N$,

$F = S$,

$\delta_n(\sigma, B_1, \ldots, B_n) = A$ for each production of the form $A \rightarrow B_1 \cdots B_n$,

$\delta_0(a) = a$ for $a \in \Sigma$.

Proposition 3.6 *Let* $G=(N, \Sigma, P, S)$ *be a wide-sense context-free grammar and* $T_A(G)$ *be the corresponding tree automaton in the sense of definition-A. Then* $S(D(G)) = L(T_A(G))$. *Furthermore, the number of states in* $T_A(G)$ *is equal to the number of nonterminal symbols in* G.

(Proof) Firstly we prove that $s \in S(D_A(G))$ iff $\delta(s)=A$ for $A \in N \cup \Sigma$. We prove it by induction on the depth of s. Suppose first that the depth of s is 0, i.e. $s=a \in \Sigma$. By the definition of $D_A(G)$ and the definition-A, $a \in D_A(G)$ iff $A=a$ iff $\delta(a)=A$. Hence $a \in S(D_A(G))$ iff $\delta(a)=A$.

Next suppose that the result holds for all trees with depth at most h. Let s be a tree of depth h+1, so that $s=\sigma(u_1,\ldots,u_n)$ for some skeletons $u_1,\ldots,u_n$ with depth at most h. Assume that $u_i \in S(D_{B_i}(G))$ for $1 \leq i \leq n$. Then

$\sigma(u_1,\ldots,u_n) \in S(D_A(G))$

iff there is a tree $A(u_1,\ldots,u_n)$ in $D_A(G)$

iff there is a production of the form $A \rightarrow B_1,\ldots,B_n$ in P of G,

by the definition of $D_A(G)$,

iff $\delta_n(\sigma, B_1,\ldots,B_n)=A$, by the definition-A,

iff $\delta_n(\sigma, \delta(u_1),\ldots,\delta(u_n))=A$, by the induction hypothesis,

iff $\delta(\sigma(u_1,\ldots,u_n))=A$.

This completes the induction and the proof of the above proposition.

Then it immediately follows from this that $s \in S(D(G))$ iff $\delta(s)=A \in F$. Hence $S(D(G)) = L(T_A(G))$.

By the definition-A, it is clear that the number of states in $T_A(G)$ is equal to the number of nonterminal symbols in G.

□

Definition-B Let $T_A=(Q, \Sigma \cup Sk, \delta, F)$ be a tree automaton for a skeletal set over Σ. The *corresponding wide-sense context-free grammar* $G(T_A)=(N, \Sigma, P, S)$ is defined with nonterminal alphabet N, start symbols S, and a finite set of productions P as follows.

$N = Q$,

$S = F$,

$P = \{\delta_n(\sigma, x_1,\ldots,x_n) \rightarrow x_1 \cdots x_n : \sigma \in Sk_n, x_1,\ldots,x_n \in Q \cup \Sigma, \text{ and } n \geq 1\}$.

Proposition 3.7 *Let* $T_A=(Q, \Sigma \cup Sk, \delta, F)$ *be a tree automaton and* $G(T_A)$ *be the corresponding context-free grammar in the sense of definition-B.*
Then $L(T_A)=S(D(G(T_A)))$.

(Proof) Firstly we prove that $\delta(s)=q$ iff $s \in S(D_q(G(T_A)))$ for $q \in Q \cup \Sigma$. We prove it by induction on the depth of s. Suppose first that the depth of s is 0, i.e. $s=a \in \Sigma$. By the definition-B and the definition of $D_A(G)$, $\delta(a)=q$ iff $q=a$ iff $a \in D_q(G(T_A))$. Hence $\delta(a)=q$ iff $a \in S(D_q(G(T_A)))$.

Next suppose that the result holds for all trees with depth at most h. Let s be a tree of depth h+1, so that $s=\sigma(u_1,\ldots,u_n)$ for some skeletons $u_1,\ldots,u_n$ with depth at most h. Assume that $\delta(u_i)=x_i$ for $1 \leq i \leq n$. Then

$\delta(\sigma(u_1,\ldots,u_n))=q$

iff $\delta_n(\sigma, \delta(u_1),\ldots,\delta(u_n))=q$

iff $\delta_n(\sigma, x_1,\ldots,x_n)=q$

iff there is a production of the form $q \rightarrow x_1,\ldots,x_n$ in $G(T_A)$, by the definition-B,

iff there is a tree $q(u_1,\ldots,u_n)$ in $D_q(G(T_A))$,

by the definition of $D_q(G(T_A))$ and the induction hypothesis,

iff $\sigma(u_1,\ldots,u_n) \in S(D_q(G(T_A)))$.

This completes the induction and the proof of the above proposition.

Then it immediately follows from this that $\delta(s)=q_f \in F$ iff $s \in S(D(G(T_A)))$. Hence $L(T_A)=S(D(G(T_A)))$.

□

4. State characterization matrix

Definition A *set of test states* S is a finite set of trees over $\Sigma \cup Sk$ with depth at least 1. The *set of transition states* is defined to be $X(S)=\{\sigma(\bar{s}) : \sigma \in Sk_i, \bar{s} \in (S \cup \Sigma)^i$, and $\sigma(\bar{s}) \notin S$ for $i \geq 1\}$. A *set of experiments* E is a finite subset of $(\Sigma \cup Sk)_\$^T$. S is called *subtree-closed* if $s \in S$ implies that all subtrees with depth at least 1 of s are elements of S. E is called *\$-prefix-closed with respect to* S if $e \in E$ except \$ implies that there exists an e' in E such that $e=e'\#\sigma(s_1,\ldots,s_{i-1},\$,s_i,\ldots,s_{n-1})$ for some $s_1,\ldots,s_{n-1}\in S\cup\Sigma$ and some i $(1\leq i\leq n)$.

Definition A *state characterization matrix* is a triple (S, E, M) where M is a matrix with labeled rows and columns such that

1) The rows are labeled with the elements of $S\cup X(S)$.

2) The columns are labeled with the elements of E.

3) Each entry of M is either 0 or 1.

4) If $s_i, s_j\in S\cup X(S)$ and $e_i, e_j\in E$ and $e_i\#s_i=e_j\#s_j$, then the (s_i, e_i) and (s_j, e_j) positions in M must have the same entry.

The *data contained in* M is $D(M)=\{(e\#s, y) : s\in S\cup X(S), e\in E$, and the entry of M is $y\in\{0, 1\}\}$. Thus we can regard D(M) as a finite function mapping $E\#(S\cup X(S))$ to $\{0, 1\}$. If s is an element of $(S\cup X(S))$, then *row(s)* denotes the finite function f from E to $\{0, 1\}$ defined by $f(e)=D(M)(e\#s)$.

Definition A state characterization matrix is called *closed* if every row(x) of transition state $x\in X(S)$ is identical to some row(s) of test states $s\in S$. A state characterization matrix is called *consistent* if whenever s_1 and s_2 are test states of S such that $row(s_1)$ is equal to $row(s_2)$, for $\sigma\in Sk_n$ and all $u_1,\ldots,u_{n-1}\in S\cup\Sigma$, $row(\sigma(u_1,\ldots,u_{i-1},s_1,u_i,\ldots,u_{n-1}))$ is equal to $row(\sigma(u_1,\ldots,u_{i-1},s_2,u_i,\ldots,u_{n-1}))$ for $0\leq i\leq n$ $(n\geq 0)$.

M	e E
s S	: .... 1 (=D(M)(e#s))
X(S)	

Figure 4.1 (S, E, M)

The ideas of the closed, consistent state characterization matrix and the algorithm using this are essentially the extensions of Angluin's ones [1] (the extension from string automata to tree automata). A sequence of following lemmas and theorems are guided by those Angluin's results. The idea of the characterization matrix is also related to the work by Gold [7].

Definition Let (S, E, M) be a closed, consistent state characterization matrix such that E contains \$. The *constructed tree automaton* $T_A(M)$ over $\Sigma \cup Sk$ from (S, E, M) is defined with state set Q, final states F, and state transition function δ as follows.

$Q = \{row(s) : s \in S\}$,

$F = \{row(s) : s \in S \text{ and } D(M)(s) = 1\}$,

$\delta_n(\sigma, row(s_1),...,row(s_n)) = row(\sigma(s_1,...,s_n))$ for $s_1,...,s_n \in S \cup \Sigma$,

$\delta_0(a) = a$ for $a \in \Sigma$,

where the function row is augmented to be $row(a) = a$ for $a \in \Sigma$.

We can see that this is a well-defined deterministic tree automaton. If s_1 and s_2 are elements of S such that $row(s_1) = row(s_2)$, then since E contains \$, $D(M)(s_1) = D(M)(\$\#s_1)$ and $D(M)(s_2) = D(M)(\$\#s_2)$ are defined and equal to each other. Hence F is well-defined. Let s_1 and s_2 be two elements of S such that $row(s_1) = row(s_2)$. Since the state characterization matrix (S, E, M) is consistent, for $u_1,...,u_{n-1} \in S \cup \Sigma$, $row(\sigma(u_1,...,u_{i-1},s_1,u_i,...,u_{n-1})) = row(\sigma(u_1,...,u_{i-1},s_2,u_i,...,u_{n-1}))$ $(0 \leq i \leq n)$, and since it is closed, this value is equal to row(s) for some s in S. Hence δ is well-defined.

Thus to distinguish two different states, the closed, consistent state characterization matrix uses the fact that for a tree automaton $T_A = (Q, \Gamma, \delta, F)$, if $\delta(t(n \leftarrow s)) \neq \delta(t(n \leftarrow s'))$ for $n \in Dom(t)$, then $\delta(s) \neq \delta(s')$. This corresponds to the contraposition of the replacement lemma.

Lemma 4.1 *Suppose that* (S, E, M) *is a closed, consistent state characterization matrix such that* S *is subtree-closed and* E *is \$-prefix-closed with respect to* S. *For the constructed tree automaton* $T_A(M)$ *and for every* s *in* $(S \cup X(S))$, $\delta(s) = row(s)$.

(Proof) We prove it by induction on the depth of s. Suppose first that the depth of s is 1, i.e., $s=\sigma(a_1,\ldots,a_n)$ for $a_1,\ldots,a_n \in \Sigma$. Since $\delta(s)=\delta_n(\sigma, \delta(a_1),\ldots,\delta(a_n))=\delta_n(\sigma, a_1,\ldots,a_n)$ $=\mathrm{row}(s)$ by the definition of δ_n, the result is clearly true. Next suppose that the result is true for all trees in $(S \cup X(S))$ with depth at most h. Let s in $(S \cup X(S))$ have depth h+1, so that $s=\sigma(s_1,\ldots,s_n)$ for some trees $s_1,\ldots,s_n$ over $\Sigma \cup Sk$ with depth at most h. Since S is subtree-closed, $s_1,\ldots,s_n$ must be in $S \cup \Sigma$. Then

$$\begin{aligned}\delta(s) &= \delta(\sigma(s_1,\ldots,s_n)) \\ &= \delta_n(\sigma, \delta(s_1),\ldots,\delta(s_n)) \\ &= \delta_n(\sigma, \mathrm{row}(s_1),\ldots,\mathrm{row}(s_n)) \\ &\qquad \text{by the induction hypothesis and the definition of } \delta_n, \\ &= \mathrm{row}(\sigma(s_1,\ldots,s_n)), \quad \text{by the definition of } \delta_n, \\ &= \mathrm{row}(s).\end{aligned}$$

□

Proposition 4.2 *Suppose that* (S, E, M) *is a closed, consistent state characterization matrix such that* S *is subtree-closed and* E *is \$-prefix-closed with respect to* S. *Then the constructed tree automaton* $T_A(M)$ *agrees with the data in* M. *That is, for every tree* s *in* $(S \cup X(S))$ *and* e *in* E, $\delta(e\#s)$ *is in* F *iff* $D(M)(e\#s)=1$.

(Proof) We prove it by induction on the depth of \$ in e. When e is \$ and s is any element of $(S \cup X(S))$, by lemma 4.1, $\delta(e\#s)=\delta(s)=\mathrm{row}(s)$. If s is in S, then by the definition of F, row(s) is in F iff $D(M)(s)=1$. If s is in X(S), then since (S, E, M) is closed, row(s) = row(s') for some s' in S, and row(s') is in F iff $D(M)(s')=1$, which is true iff $D(M)(s)=1$.

Next suppose that the result holds for all e in E where the depth of \$ is at most h. Let e be an element of E where the depth of \$ is h+1. Since E is \$-prefix-closed with respect to S, $e=e'\#\sigma(s_1,\ldots,s_{i-1},\$,s_i,\ldots,s_{n-1})$ for some $s_1,\ldots,s_{n-1} \in S \cup \Sigma$, some i $(1 \le i \le n)$ and some e' in E where the depth of \$ is h. For any element s of $(S \cup X(S))$, since (S, E, M) is closed, there is an element s' in S such that row(s) = row(s'). Then

$\delta(\sigma(s_1,\ldots,s_{i-1},s,s_i,\ldots,s_{n-1}))$

$= \delta_n(\sigma, \delta(s_1),\ldots,\delta(s_{i-1}),\delta(s),\delta(s_i),\ldots,\delta(s_{n-1}))$

$= \delta_n(\sigma, row(s_1),\ldots,row(s_{i-1}),row(s),row(s_i),\ldots,row(s_{n-1}))$, by lemma 4.1,

$= \delta_n(\sigma, row(s_1),\ldots,row(s_{i-1}),row(s'),row(s_i),\ldots,row(s_{n-1}))$

since $row(s) = row(s')$,

$= row(\sigma(s_1,\ldots,s_{i-1},s',s_i,\ldots,s_{n-1}))$, by the definition of δ_n,

$= \delta(\sigma(s_1,\ldots,s_{i-1},s',s_i,\ldots,s_{n-1}))$, by lemma 4.1.

Therefore

$\delta(e\#s) = \delta(e'\#\sigma(s_1,\ldots,s_{i-1},\$,s_i,\ldots,s_{n-1})\#s)$

$= \delta(e'\#\sigma(s_1,\ldots,s_{i-1},s,s_i,\ldots,s_{n-1}))$

$= \delta(e'\#\sigma(s_1,\ldots,s_{i-1},s',s_i,\ldots,s_{n-1}))$,

by the above and the replacement lemma.

By the induction hypothesis,

$\delta(e'\#\sigma(s_1,\ldots,s_{i-1},s',s_i,\ldots,s_{n-1}))$ is in F iff $D(M)(e'\#\sigma(s_1,\ldots,s_{i-1},s',s_i,\ldots,s_{n-1})) = 1$.

Since $row(s) = row(s')$,

$D(M)(e'\#\sigma(s_1,\ldots,s_{i-1},s',s_i,\ldots,s_{n-1})) = D(M)(e'\#\sigma(s_1,\ldots,s_{i-1},s,s_i,\ldots,s_{n-1}))$,

and since $e'\#\sigma(s_1,\ldots,s_{i-1},\$, s_i,\ldots,s_{n-1}) = e$ is in E,

$D(M)(e'\#\sigma(s_1,\ldots,s_{i-1},s,s_i,\ldots,s_{n-1})) = D(M)(e\#s)$.

Hence $\delta(e\#s)$ is in F iff $D(M)(e\#s) = 1$.

□

For the proof of the following theorem, for a tree automaton $T_A = (Q, \Gamma, \delta, F)$ we extend δ to $(\Gamma \cup Q)^T$ by letting : $\delta(q) = q$ for $q \in Q$, where Q is considered as a set of constant symbols. In this definition, if $q = \delta(s)$ for $q \in Q$ and $s \in \Gamma^T$, then $\delta(t(x \leftarrow q)) = \delta(t(x \leftarrow s))$ for $t \in \Gamma^T$ and $x \in Dom(t)$.

Proposition 4.3 *Suppose that* (S, E, M) *is a closed, consistent state characterization matrix such that* S *is subtree-closed and* E *is \$-prefix-closed with respect to* S. *Suppose that the constructed tree automaton* $T_A(M) = (Q, \Sigma \cup Sk, \delta, F)$ *from* (S, E, M) *has* n *states. If* $T_A' = (Q', \Sigma \cup Sk, \delta', F')$ *is any tree automaton which agrees with the data in* M *that has* n *or fewer states, then* T_A' *is isomorphic to* $T_A(M)$.

567

(Proof) We prove it by exhibiting an isomorphism ϕ. First define, for each q' in Q', row(q') to be the finite function f from E to {0, 1} such that f(e) = 1 iff δ'(e#q') is in F'. Since T_A' agrees with the data in M, for each s in (S∪X(S)) and each e in E, δ'(e#s) is in F' iff D(M)(e#s) = 1, so row(δ'(s)) is equal to row(s) in (S, E, M). Hence as s ranges over all of S, row(δ'(s)) ranges over all the elements of Q, so T_A must have at least n states, i.e., it must have exactly n states. Thus, for each s in S there is a unique q' in Q' such that row(s) = row(q'), namely, δ'(s).

Next we define for each s in S, ϕ(row(s)) to be δ'(s) and for a∈Σ, ϕ(a) = a. This mapping is one-to-one and onto. We must verify that it preserves the transition function, and that it carries F to F'. For each $s_1,...,s_n$ in S∪Σ and $\sigma \in Sk_n$, let s be an element of S such that row($\sigma(s_1,...,s_n)$) = row(s). Then

$$\phi(\delta_n(\sigma, row(s_1),...,row(s_n))) = \phi(row(\sigma(s_1,...,s_n)))$$
$$= \phi(row(s))$$
$$= \delta'(s).$$

Also,

$$\delta'_n(\sigma, \phi(row(s_1)),...,\phi(row(s_n))) = \delta'_n(\sigma, \delta'(s_1),...,\delta'(s_n))$$
$$= \delta'(\sigma(s_1,...,s_n)).$$

Since δ'(s) and $\delta'(\sigma(s_1,...,s_n))$ have identical row values, namely row(s) and row($\sigma(s_1,...,s_n)$), they must be the same state of T_A'.

Hence $\phi(\delta_n(\sigma, row(s_1),...,row(s_n))) = \delta'_n(\sigma, \phi(row(s_1)),...,\phi(row(s_n)))$ for all $s_1,...,s_n$ in S∪Σ and $\sigma \in Sk_n$. Lastly, since if s in S has row(s) in F, then D(M)(s) = 1, so since ϕ(row(s)) is mapped to a state q' with row(q') = row(s), it must be that q' is in F'. Conversely, if row(s) is mapped to a state q' in F', then since row(q') = row(s), D(M)(s) = 1, so row(s) is in F. Thus ϕ maps F to F'. So we conclude that the mapping ϕ is an isomorphism of T_A(M) and T_A'.

□

5. Inductive inference algorithm for context-free grammar

Now we describe an inference algorithm which efficiently infers an unknown context-free grammar G_U. We assume that a finite alphabet Σ which the G_U is defined over and a skeletal alphabet Sk for the G_U are given.

5.1 Algorithm 1

Definition (construction of a context-free grammar) Let (S, E, M) be a closed, consistent state characterization matrix such that E contains \$. The *constructed wide-sense context-free grammar* $G(M)=(N, \Sigma, P, S)$ from (S, E, M) is defined with nonterminal alphabet N, start symbols $S \subseteq N$, and a finite set of productions P as follows.

$N = \{row(s) : s \in S\}$,

$S = \{row(s) : s \in S \text{ and } D(M)(s) = 1\}$,

$P = \{row(\sigma(s_1,\ldots,s_n)) \rightarrow row(s_1)\cdots row(s_n)\}$,

where the function row is augmented to be $row(a) = a$ for $a \in \Sigma$.

Let T_D be a tree automaton over $\Sigma \cup Sk$ which consists of only one state q_d, and state transition function such that $\delta_n(\sigma, q_d,\ldots,q_d) = q_d$ with no final state (i.e. $F = \varnothing$). Clearly, $L(T_D) = \varnothing$.

(Algorithm 1 of inductive inference for context-free grammar)

Input : An oracle EX() for the set of examples of the skeletal descriptions of the unknown context-free grammar G_U, i.e. examples of $+s$ for $s \in S(D(G_U))$ and $-s$ for $s \in (\Sigma \cup Sk)^T - S(D(G_U))$,

An oracle MEMBER(s) on a skeleton s as input for a membership query to output 1 or 0 according to whether s is a skeletal description of a derivation tree of G_U from S, i.e. $s \in S(D(G_U))$,

Output : A sequence of conjectures of context-free grammar,

Procedure:

```
S := ∅; E := {$};
TA := T_D; CFG := ∅; Examples := ∅;
do forever
   add an example EX() to Examples;
   while there is a negative example −s∈Examples which TA accepts s or
         there is a positive example +s∈Examples which TA does not accept s;
      add s and all its subtrees except constants to S;
      extend (S, E, M) to E#(S∪X(S)) using MEMBER;
      repeat
         if (S, E, M) is not consistent
            then find s1 and s2 in S, u1,...,u_{n-1}∈S∪Σ, e∈E, and i (1≦i≦n) such that
                 row(s1) is equal to row(s2) and
                 D(e#σ(u1,...,u_{i-1},s1,u_i,...,u_{n-1}))≠D(e#σ(u1,...,u_{i-1},s2,u_i,...,u_{n-1}));
            add e#σ(u1,...,u_{i-1},$,u_i,...,u_{n-1}) to E;
            extend (S, E, M) to E#(S∪X(S)) using MEMBER;
         if (S, E, M) is not closed;
            then find σ(s̄)∈X(S) such that row(σ(s̄)) is different from row(s)
                 for all s∈S;
            add σ(s̄) to S;
            extend (S, E, M) to E#(S∪X(S)) using MEMBER;
      until (S, E, M) is closed and consistent;
      TA := T_A(M);
      CFG := G(M);
   end;
   output CFG;
end.
```

In the above algorithm, the operation of "extend (S, E, M) to $E\#(S \cup X(S))$ using MEMBER" is the operation to extend D(M) by asking membership queries MEMBER(s) for missing elements s. We call an example s presented by the oracle EX a *counter-example* when the last conjecture $T_A(M)$ does not agree with s, i.e. $T_A(M)$ accepts a negative example $-s$ or does not accept a positive example $+s$.

5.2 Inference of parser

By replacing the construction of a context-free grammar with the following construction of a parser in the algorithm 1, we will get an inductive inference algorithm which infers a parser written in PROLOG.

Definition (construction of a parser) Let (S, E, M) be a closed, consistent state characterization matrix such that E contains \$. The *constructed parsing Prolog program* PARSER(M) using difference-lists from (S, E, M) is defined with the predicate set Predicate, the finite set of function symbols Function, the calling predicate sentence(T,X,X'), and the finite set of clauses PARSER(M) as follows :

$\text{Predicate} = \{\text{phrase}_{row(s)}(T,X,X') : s \in S\} \cup \{\text{terminal}_a(a,[a|X],X) : a \in \Sigma\}$,

$\text{Function} = \{\text{phrase}_{row(s)} : s \in S\}$,

$\text{PARSER}(M) =$

$\{\text{sentence}(T,X_0,X_1) :- \text{phrase}_{row(s)}(T,X_0,X_1) \ : s \in S \text{ and } D(M)(s) = 1\}$

$\cup\{\text{phrase}_{row(\sigma(s_1,\ldots,s_n))}(\text{phrase}_{row(\sigma(s_1,\ldots,s_n))}(T_1,\ldots,T_n),X_0,X_n) :-$

$R_1(T_1,X_0,X_1),\ldots,R_n(T_n,X_{n-1},X_n).$

$: s_i \in S \cup \Sigma, \sigma \in Sk_n$, and

$R_i = \text{phrase}_{row(s_i)}$ if $s_i \in S$ and $R_i = \text{terminal}_a$ if $s_i = a \in \Sigma \ (1 \leq i \leq n)\}$

$\cup\{\text{terminal}_a(a, [a|X],X). \ : a \in \Sigma\}$.

6. Correctness and complexity

To see that the algorithm 1 is correct, i.e. the algorithm 1 identifies a context-free grammar G in the limit such that $L(G) = L(G_U)$ for the unknown context-free grammar G_U, it is enough for us to show that the constructed state characterization matrix (S, E, M) during the running of the algorithm 1 is a closed, consistent one such that S is subtree-closed and E is \$-prefix-closed with respect to S, and that the while loop of the algorithm 1 is executed at most in a finite time during the running of the algorithm 1.

Lemma 6.1 *Let* (S, E, M) *be a state characterization matrix such that* S *is subtree-closed and* E *is \$-prefix-closed with respect to* S. *Let* n *be the number of different values of* row(s) *for* s *in* S. *Any deterministic tree automaton which agrees with the data in* M *must have at least* n *states.*

(Proof) Let $T_A = (Q, \Gamma, \delta, F)$ be a deterministic tree automaton which agrees with the data in M. Suppose that s_1 and s_2 are elements of S such that row(s_1) and row(s_2) are distinct. Then there exists e in E such that $D(M)(e\#s_1) \neq D(M)(e\#s_2)$. Since T_A agrees with the data in M, exactly one of $\delta(e\#s_1)$ and $\delta(e\#s_2)$ is in F. Thus $\delta(s_1)$ and $\delta(s_2)$ must be distinct states because T_A is deterministic. Since $\delta(s)$ takes on at least n different values as s ranges over S, T_A must have at least n states.

□

Lemma 6.2 *The while loop of the algorithm 1 is executed at most in a finite time during the running of the algorithm 1.*

(Proof) Let n be the number of states in the minimum state deterministic tree automaton T_{Am} for S(D(G_U)) of the unknown context-free grammar G_U. Firstly we will show that whenever a state characterization matrix (S, E, M) is not consistent or not closed, the number of distinct values row(s) for s in S must increase. If (S, E, M) is not consistent, then since two previously equal row values, row(s_1) and row(s_2), are no longer equal after E is augmented, the number of distinct values row(s) for s in S must increase by at least one. If (S, E, M) is not closed and a tree $\sigma(\bar{s})$ is added to S, then since row($\sigma(\bar{s})$) is different from row(s) for all s in S before S is augmented, the number of distinct values row(s) must increase by at least one.

Next we will show that whenever a tree s and all its subtrees are added to S because $T_A(M)$ does not agree with t, the next conjecture $T_A(M')$ must have at least one more state than $T_A(M)$. If a conjecture $T_A(M)$ is found to be incorrect by the example t, then since $T_A(M')$ is correct for the data in M and inequivalent to $T_A(M)$ (since $T_A(M')$ is correct for the data t and so they disagree on t), by proposition 4.3, $T_A(M')$ must have at least one more state.

Then by these and lemma 6.1, (S, E, M) can be not consistent or not closed at most $n-1$ times and a counter-example is added to S at most n times during the running of the algorithm 1. Thus whenever the condition of the while loop becomes true, the algorithm 1 eventually makes a next conjecture in finite time, and the condition of the while loop becomes true at most n times. Therefore, the while loop is executed at most in a finite time.

□

By the above result, it follows that the algorithm 1 makes at most a finite number of conjectures.

Lemma 6.3 *The conjectures which the algorithm 1 makes are correct for the facts known by the oracles* EX *and* MEMBER.

(Proof) We will show that each state characterization matrix (S, E, M) during the running of the algorithm 1 is a closed, consistent one such that S is subtree-closed and E is \$-prefix-closed with respect to S. In the algorithm 1, there are three operations which extend the row or the column of (S, E, M). When t and all its subtrees are added to S, S obviously remains subtree-closed. If (S, E, M) is not consistent, then for some $\sigma \in Sk_n$, $u_1,\ldots,u_{n-1} \in S \cup \Sigma$, $e \in E$, and i ($1 \leq i \leq n$), $e\#\sigma(u_1,\ldots,u_{i-1},\$,u_i,\ldots,u_{n-1})$ is added to E. In this case, E remains \$-prefix-closed with respect to S. If (S, E, M) is not closed, then for some $\bar{s} \in (S \cup \Sigma)^n$ and $\sigma \in Sk_n$, $\sigma(\bar{s})$ is added to S. In this case, S remains subtree-closed. Since the repeat loop is repeated as long as (S, E, M) is not closed and consistent, by lemma 6.2, each constructed (S, E, M) must eventually be closed and consistent. Thus each constructed (S, E, M) during the running of the algorithm 1 is a closed, consistent one such that S is subtree-closed and E is \$-prefix-closed with respect to S. Then by proposition 4.2 and proposition 3.7, the conjectures of wide-sense context-free grammar which the algorithm 1 makes are correct for the facts known by the oracles EX and MEMBER.

□

In the conjectures of context-free grammar of the algorithm 1, we can effectively detect and eliminate the nonterminal which cannot be derived from S (that corresponds to dead state) and all productions which include it. By adding this operation on the conjectures to the algorithm 1, we conclude the following theorem.

Theorem 6.4 *Let* G_U *be an unknown context-free grammar. Given the oracles* EX *and* MEMBER *for* G_U*, the algorithm 1 identifies in the limit a minimum nonterminal wide-sense context-free grammar* CFG *such that* $L(CFG)=L(G_U)$, CFG *is structurally equivalent to* G_U *and no two productions in* P *have the same right side.*

(Proof) By lemma 6.2, 6.3, proposition 3.6, and 3.7.

□

The above theorem states that for a sequence of conjectures $CFG_1, CFG_2, CFG_3, \ldots$ by the algorithm 1, there exists a finite time t such that for all t'>t, $CFG_{t'}=CFG_t$ and $L([CFG_t])=L([G_U])$. In [4], this type of identification is called *structural identification in the limit.*

In [9], the grammars which have unique right hand sides of the productions are called *invertible grammars*, which is one of the normal forms for context-free grammars. Invertible grammars allow the process of bottom-up parsing to be made simply.

Next we will analyse the time complexity of the algorithm 1. By lemma 6.2, the while loop of the algorithm 1 is executed at most in a finite time. Then how much time does the while loop consume during the running of the algorithm 1. That depends partly on the size of the examples t presented by the oracle EX. We will analyze the running time of the while loop as a function of n, the number of states in the minimum tree automaton for $S(D(G_U))$ of the unknown context-free grammar G_U, and m, the maximum size of any counter-examples presented by EX during the running of the algorithm 1, where the *size* of an example is the number of symbols in its textual representation. We will show that its running time is bounded by a polynomial in m and n. Let k be the cardinality of the skeletal alphabet Sk (that is the number of distinct ranks of the symbol σ) and d be the maximum rank of the symbol σ in Sk. We may assume $d \geqq 1$.

Whenever (S, E, M) is discovered to be not closed, one element is added to S. Whenever (S, E, M) is discovered to be not consistent, one element is added to E. For each counter-example t of size at most m presented by the oracle EX, at most m subtrees are added to S. Since the state characterization matrix is discovered to be not consistent at most $n-1$ times, the total number of trees in E cannot exceed n. Since the state characterization matrix is discovered to be not closed at most $n-1$ times, and since there can be at most n counter-examples, the total number of trees in S cannot exceed $n+mn$. Thus, the maximum cardinality of $E\#(S \cup X(S))$ is at most

$$n((n+mn)+k(n+mn)^d) = O(m^d n^{d+1}).$$

Now we consider the operations in the while loop executed by the algorithm 1. Checking the state characterization matrix to be closed and consistent can be done in time polynomial in the size of the matrix and must be done at most n times. Adding a tree to S or E requires at most $O(m^d n^d)$ membership queries to extend the matrix. When the state characterization matrix is closed and consistent, $T_A(G)$ and G(M) may be constructed in time polynomial in the size of the matrix, and this must be done at most n times. A counter-example requires the addition of at most m subtrees to S, and this can be also happen at most n times.

Therefore, *the total time which the while loop consumes during the running of the algorithm 1 can be bounded by a polynomial function of* m *and* n.

On the other hand, the check whether a conjecture agrees with an example, i.e. TA accepts s or not, in the condition of the while loop is decidable and is performed in steps of the example's size. Then by the above result, we can conclude that *the algorithm 1 infers a conjecture of context-free grammar and requests a new example in time polynomial in* l, m' *and* n *after the last example has been added*, where l is the number of examples known at the time of the request and m' is the maximum size of those l known examples.

7. An example

In inferring context-free grammar from their structural descriptions, given a set of derivation trees from the unknown grammar with all nonterminal labels erased, the problem is to reconstruct the nonterminal labels. Our algorithm 1 distinguishes internal nodes of structural descriptions of the unknown grammar by using a set of experiments E and reconstruct the nonterminal labels.

Suppose that the unknown grammar is the following context-free grammar $G_U = (N, \Sigma, P, S)$ which generates the set of all valid arithmetic expressions involving a variable "v", the operations of multiplication "×" and addition "+", and the parentheses "[" and "]":

$N = \{S, E, F\}$,

$\Sigma = \{v, \times, +, [,]\}$,

$P = \{ S \rightarrow E$

$E \rightarrow F$

$E \rightarrow F + E$

$F \rightarrow v$

$F \rightarrow v \times F$

$F \rightarrow [E] \}$.

Firstly, we give the algorithm 1 an example

$$\sigma(\sigma(\sigma(v, \times, \sigma([, \sigma(\sigma(v), +, \sigma(\sigma(v))),]))))$$

which is a structural description of a derivation tree

$$S(E(F(v, \times, F([, E(F(v), +, E(F(v))),]))))$$

for a sentence $v \times [v+v]$ assigned by G_U.

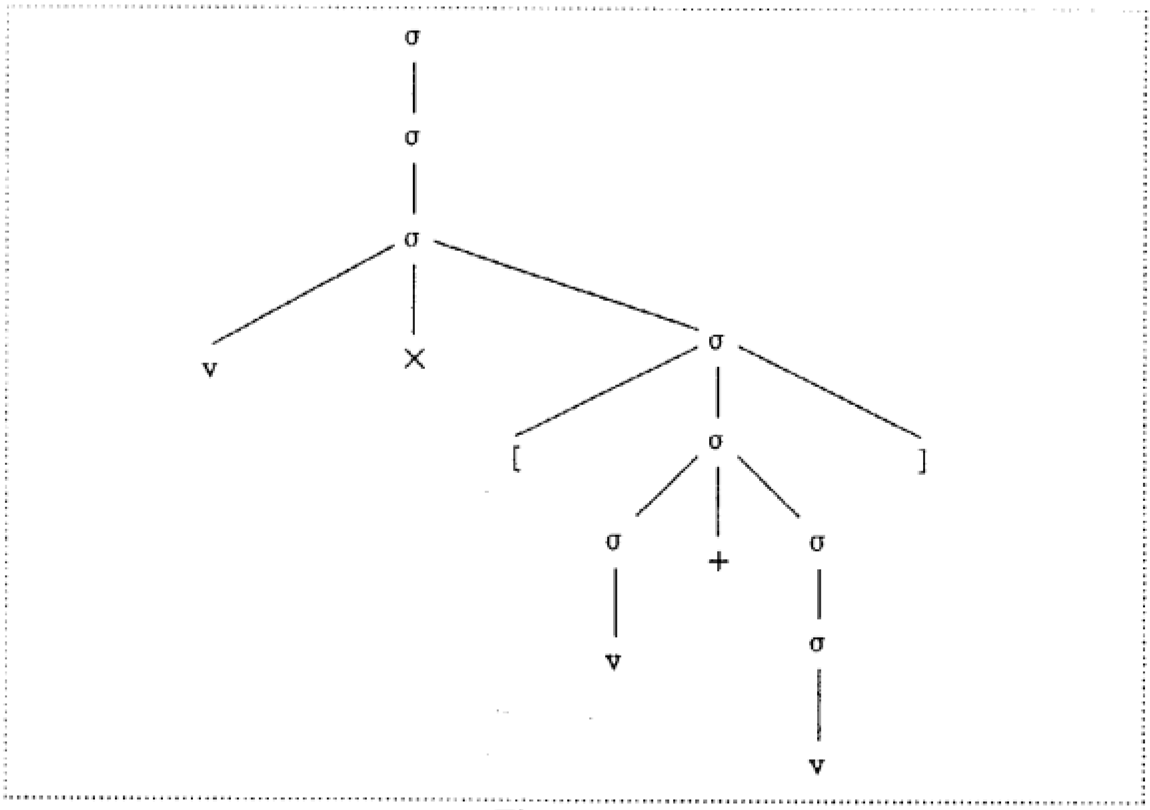

Figure 7.1

The algorithm 1 adds all subtrees of it to S and divides them into two parts (i.e. row(s)=0 and row(s')=1) by asking membership queries of them. Thus internal nodes of the structural description are labeled 0 or 1 of two row values, where $E=\{\$\}$, shown in Figure 7.2.

Next the algorithm 1 tries to make a closed, consistent state characterization matrix by asking membership queries. In this process, the algorithm 1 discovers the matrix to be not consistent once, and so it adds the experiment $\sigma(\$)$ to E. Then the algorithm 1 makes a closed, consistent state characterization matrix and outputs the first conjecture of context-free grammar, shown in Figure 7.3.

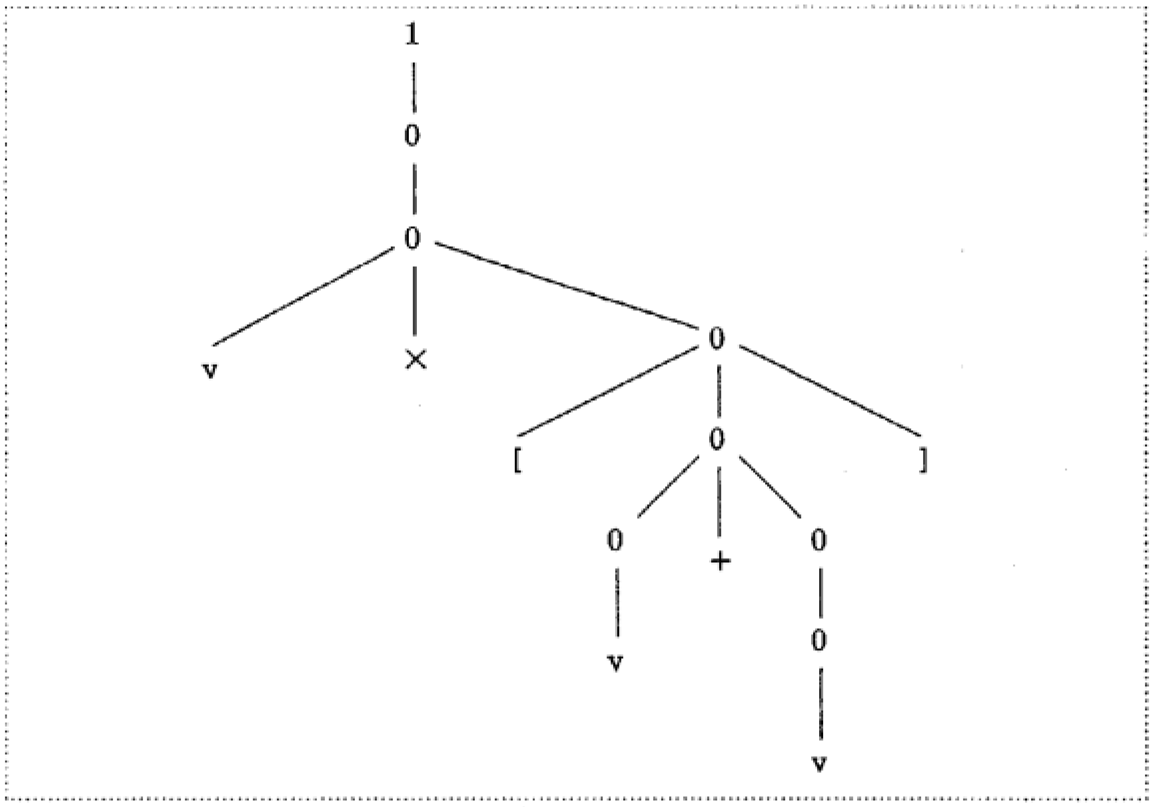

Figure 7.2

(The first conjecture $G_1 = ((N_1, \Sigma, P_1, S_1))$

N_1 = { <10>, <01>, <00> },

S_1 = { <10> },

P_1 = { <00> → v

<01> → <00>

<01> → <00> + <01>

<00> → [<01>]

<00> → v × <00>

<10> → <01>

<00> → <10>

<00> → v × <01>

. . . },

Figure 7.3

(State characterization matrix)

		"E"	
	M	$	σ($)
"S"	σ(v)	0	0
	σ(σ(v))	0	1
	σ(σ(v), + ,σ(σ(v)))	0	1
	σ([,σ(σ(v), + ,σ(σ(v))),])	0	0
	σ(v, × ,σ([,σ(σ(v), + ,σ(σ(v))),]))	0	0
	σ(σ(v, × ,σ([,σ(σ(v), + ,σ(σ(v))),])))	0	1
	σ(σ(σ(v, × ,σ([,σ(σ(v), + ,σ(σ(v))),]))))	1	0
"X(S)"	. . .	. . .	. . .

Figure 7.4

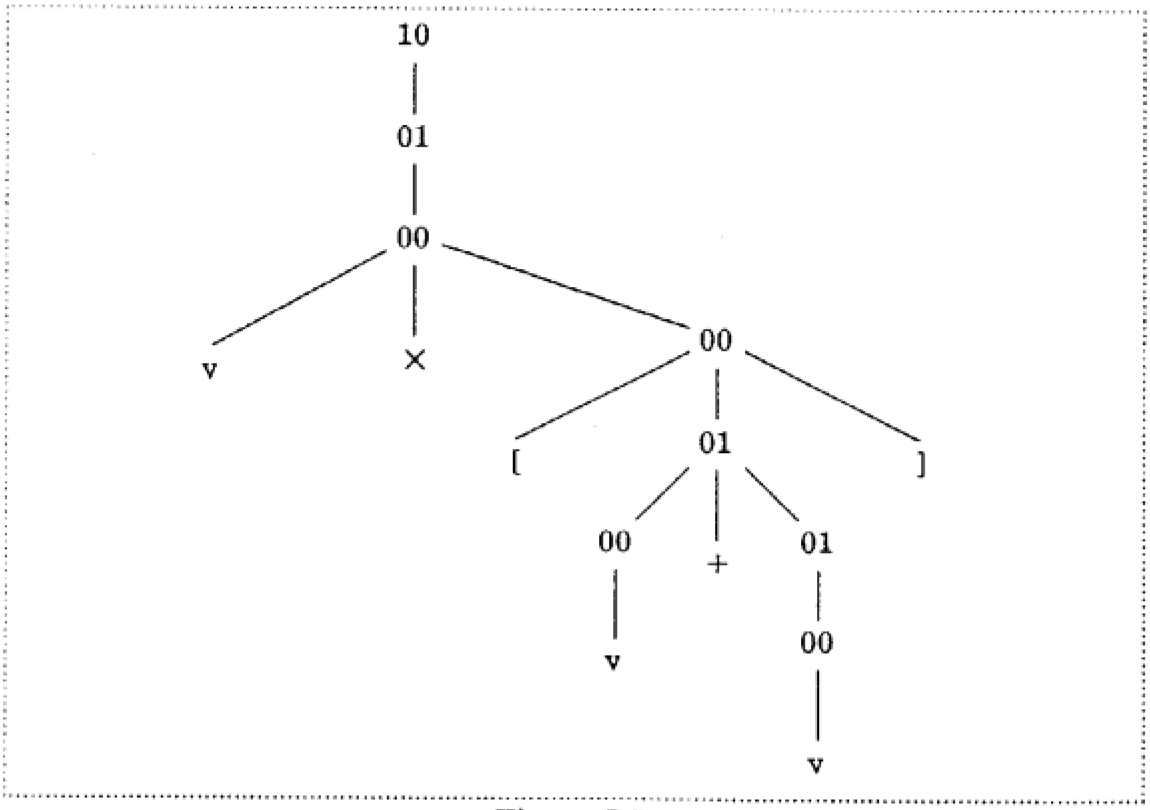

Figure 7.5

However G_1 is not correct for G_U, so we give a counter-example σ(σ(σ(σ(σ(σ(v)))))), which is in $S(D(G_1))$ but not in $S(D(G_U))$, to the algorithm 1. The algorithm 1 eventually makes other closed, consistent state characterization matrix, shown in Figure 7.6, and outputs the second conjecture of context-free grammar. This conjecture is a correct grammar for G_U, and furthermore structurally equivalent to G_U. The reduced version of it is shown in Figure 7.7 by eliminating the meaningless nonterminal and all productions including it.

(State characterization matrix)

		"E"		
	M	\$	σ(\$)	σ(σ(\$, +, σ(σ(v))))
"S"	σ(v)	0	0	1
	σ(σ(v))	0	1	0
	σ(σ(v), +, σ(σ(v)))	0	1	0
	σ([, σ(σ(v), +, σ(σ(v))),])	0	0	1
	σ(v, ×, σ([, σ(σ(v), +, σ(σ(v))),]))	0	0	1
	σ(σ(v, ×, σ([, σ(σ(v), +, σ(σ(v))),])))	0	1	0
	σ(σ(σ(v, ×, σ([, σ(σ(v), +, σ(σ(v))),]))))	1	0	0
	σ(σ(σ(v)))	1	0	0
	σ(σ(σ(σ(v))))	0	0	0
	σ(σ(σ(σ(σ(v)))))	0	0	0
	σ(σ(σ(σ(σ(σ(v))))))	0	0	0
"X(S)"	. . .	. . .	. . .	. . .

Figure 7.6

(The second conjecture $G_2 = ((N_2, \Sigma, P_2, S_2))$

N_1 = { <100>, <010>, <001> },

S_1 = { <100> },

P_1 = { <001> → v

<010> → <001>

<010> → <001> + <010>

<001> → [<010>]

<001> → v × <001>

<100> → <010> }.

Figure 7.7

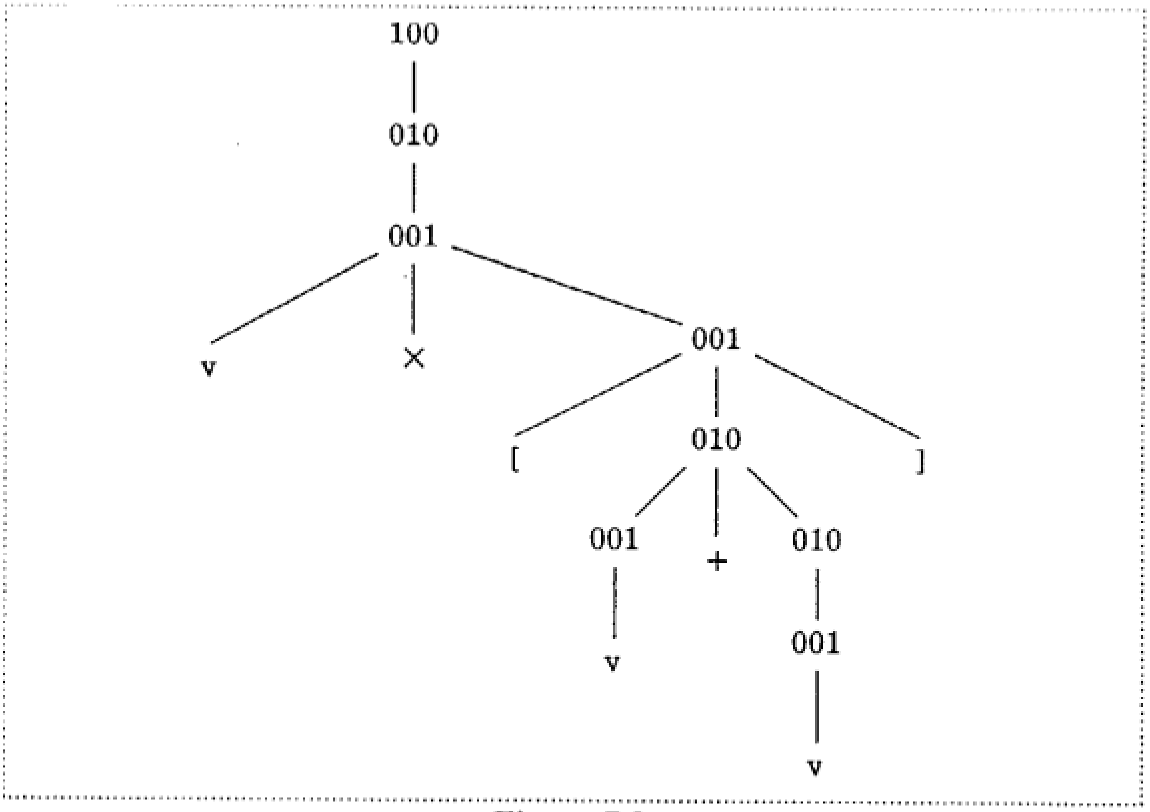

Figure 7.8

8. More efficient method

In the algorithm 1, a (minimal) tree automaton which corresponds to the unknown context-free grammar is identified in the limit. However, since the algorithm will have all structural descriptions of the unknown grammar which corresponds to all derivation trees with non-labeled nodes for nonterminals, it is enough for us to identify the nonterminal labels. Thus with the nonterminal labels identified and structural descriptions of the unknown grammar, we can easily reconstruct the productions of the grammar and the grammar itself. Then a more efficient inference algorithm can be obtained using the following smaller characterization matrix without X(S) part than the previous state characterization matrix.

Definition Let S be a finite set of trees over $\Sigma \cup Sk$ with depth at least 1 and E be a finite subset of $(\Sigma \cup Sk)_\$^T$. A *nonterminal characterization matrix* is a triple (S, E, M_N), where M_N is a matrix with labeled rows and columns such that

1) The rows are labeled with the elements of S.

2) The columns are labeled with the elements of E.

3) Each entry of M_N is either 0 or 1.

4) If $s_i, s_j \in S$ and $e_i, e_j \in E$ and $e_i \# s_i = e_j \# s_j$, then the (s_i, e_i) and (s_j, e_j) positions in M_N must have the same entry.

The *data contained in* M_N is $D(M_N) = \{(e\#s, y) : s \in S, e \in E$, and the entry of M_N is $y \in \{0, 1\}\}$.

Definition Let G be a context-free grammar. A nonterminal characterization matrix is called *consistent with respect to* G if the data in M_N agree with G (i.e. $D(M_N)(s) = 1$ iff $s \in S(D(G))$), and whenever s_1 and s_2 are elements of S such that $row(s_1)$ is equal to $row(s_2)$, then for all $s \in S$ such that $s = \sigma(u_1, \ldots, u_{i-1}, s_1, u_i, \ldots, u_{n-1})$ for some $u_1, \ldots, u_{n-1} \in S \cup \Sigma$ and i, and for all $e \in E$, $e\#s \in S(D(G))$ iff $e\#\sigma(u_1, \ldots, u_{i-1}, s_2, u_i, \ldots, u_{n-1}) \in S(D(G))$.

M_N	e E
s S	: ⋯⋯⋯ 1 $(=D(M_N)(e\#s))$

Figure 8.1 (S, E, M_N)

Definition (construction of a context-free grammar) Let (S, E, M_N) be a nonterminal characterization matrix such that E contains \$. The *constructed wide-sense context-free grammar* $G(M_N) = (N, \Sigma, P, S)$ from (S, E, M_N) is defined with nonterminal alphabet N, start symbols $F \subseteq N$, and a finite set of productions P as follows.

$N = \{row(s) : s \in S\}$,

$S = \{row(s) : s \in S \text{ and } D(M_N)(s) = 1\}$,

$P = \{row(\sigma(s_1,...,s_n)) \rightarrow row(s_1) \cdots row(s_n)\}$,

where the function row is augmented to be $row(a) = a$ for $a \in \Sigma$.

Now we describe a more efficient inference algorithm than the algorithm 1, which uses the nonterminal characterization matrix.

(Algorithm 2 of inductive inference for context-free grammar)

Input : An oracle EX() for the set of examples of the skeletal descriptions of the unknown context-free grammar G_U, i.e. examples of $+s$ for $s \in S(D(G_U))$ and $-s$ for $s \in (\Sigma \cup Sk)^T - S(D(G_U))$,

An oracle MEMBER(s) on a skeleton s as input for a membership query to output 1 or 0 according to whether s is a skeletal description of a derivation tree of G_U from S, i.e. $s \in S(D(G_U))$,

Output : A sequence of conjectures of context-free grammar,

Procedure :

$S := \varnothing$; $E := \{\$\}$;

CFG $:= \varnothing$; Examples $:= \varnothing$;

do forever

add an example EX() to Examples;

while there is a negative example $-s \in$ Examples such that $s \in S(D(CFG))$ or

there is a positive example $+s \in$ Examples $s \notin S(D(CFG))$;

add s and all its subtrees except constants to S;

extend (S, E, M_N) to E#S using MEMBER;

repeat

if (S, E, M_N) is not consistent with respect to G_U

then find s_1, s_2 and s in S, and e in E such that

$s = \sigma(u_1,...,u_{i-1},s_1,u_i,...,u_{n-1})$ for some $u_1,...,u_{n-1} \in S \cup \Sigma$ and i

and MEMBER(e#s) $\neq$ MEMBER(e#$\sigma(u_1,...,u_{i-1},s_2,u_i,...,u_{n-1})$);

add e#$\sigma(u_1,...,u_{i-1},\$,u_i,...,u_{n-1})$ to E;

extend (S, E, M) to E#S using MEMBER;

until (S, E, M_N) is consistent with respect to G_U

CFG := G(M_N);

end;

output CFG;

end.

We will not state the correctness of the algorithm 2. However we will analyse the time complexity of it. Let l, m, m', n, and d be the parameters defined in the section 6. Since the nonterminal characterization matrix is discovered to be not consistent with respect to G_U at most $n-1$ times, the total number of trees in E cannot exceed n, and since there can be at most n counter-examples, the total number of trees in S cannot exceed mn. Thus, the maximum cardinality of E#S is at most mn^2. Checking the nonterminal characterization matrix to be consistent with respect to G_U requires at most m^2n^3 membership queries and can be done in $O(m^3n^4)$ time and must be done at most n times. Adding a tree to S or E requires at most mn membership queries. G(M_N) may be constructed in time polynomial in the size of the matrix. A counter-example requires the addition of at most m subtrees to S. Thus, *the algorithm 2 infers a conjecture of context-free grammar and requests a new example in time polynomial in* l, m' *and* n *after the last example has been added.* The point is that the time is bounded by a polynomial in l, m', and n, and is no longer exponential in d. That is, the time is bounded by a polynomial of a fixed constant k ($k \leq 8$) independent of d or the unknown grammar.

9. Discussions

We remark on related work. A literature [4] is closely related, as it describes a constructive method for inferring a context-free grammar from bracketed examples, i.e. examples of structural descriptions of the language. In the paper, Crespi-Reghizzi introduces the notion of *structural identification in the limit.* His inference algorithm is to infer from positive data so that it is only to infer a class of context-free grammars, called *free operator precedence grammars.* Levy and Joshi [11] show a theoretical framework for grammatical inference in terms of structural descriptions and have inspired our work. They show that a finite set of skeletons can characterize a context-free language so that we need only construct a tree automaton which recognizes the set of skeletons. Fass [6] presents an algorithmic solution to the inference problem of context-free languages from their structured sentences based on the theory of Levy and Joshi. His solution, however, only gives a theoretical basis for grammatical inference and his algorithm is still impractical or inefficient. Furthermore, these works are not formally discussed in the concept of *identification in the limit* defined by Gold [8]. In another sense, Berger and Pair [2] are closely related, as it describes inference for tree languages, called *regular bilanguages.* Their study is a general approach in an abstract setting.

We consider an application of our algorithm to an inference from sentences (not structural ones). Our algorithm needs the structural information of the unknown grammar. However, if the algorithm automatically constructs the structure of sentences, then it could infer the unknown language only from their sentences. As we have seen in section 3, the structure of the language can be described by means of a parenthesis grammar. Then our algorithm can infer the class of parenthesis languages only from their sentences if the information about the symbols which play roles of parentheses is given to the algorithm, because the structural information can be obtained from sentences of a parenthesis language. Furthermore, our algorithm can infer only from sentences the class of generalized parenthesis languages in the sense of [15], which is a proper superclass of the parenthesis languages and may be able to define the most part of the programming languages.

As Crespi-Reghizzi et al. [5] suggest, grammatical inference may be useful in specifying programming languages. A practical application of our algorithm is designing programming languages or synthesis of compiler, because the structure or syntax of programming languages is usually defined by means of a context-free grammar. As in [5], the definition of structure and the definition of meaning should be interconnected since structural orderings are an aid for interpreting a sentence. Thus in inferring a programming language, a grammar inferred for the language should be constructed such that it not only generates correctly sentences but also assigns to each sentence a structure required by the designer. Then our approach will provide an effective method for the process of programming language design.

Acknowledgements

The author would like to thank Dr. T.Kitagawa, the president of IIAS-SIS, Dr. H.Enomoto, the director of IIAS-SIS, for giving him the opportunity to pursue this work and helping him. He is deeply grateful to Dr. T.Yokomori for reading the draft and giving him many valuable comments. Discussions with the colleagues Y.Takada and H.Ishizaka were also very fruitful.

This is part of the work in the major R&D of the Fifth Generation Computer Project, conducted under program set up by MITI.

References

[1] Angluin,D., Learning regular sets from queries and counter-examples, *Yale DCS TR-464*, 1986, to appear in *Information and Computation*.

[2] Berger,J., Pair,C., Inference for regular bilanguages, *J. Comput. Sys. Sci. 16* (1978), 100-122.

[3] Courcelle,B., Fundamental properties of infinite trees, *Theor. Comput. Sci. 25* (1983), 95-169.

[4] Crespi-Reghizzi,S., An effective model for grammar inference, in *Information Processing 71*, Gilchrist,B., Ed., Elsevier North-Holland (1972), 524-529.

[5] Crespi-Reghizzi,S., Melkanoff,M.A., and Lichten,L., The use of grammatical inference for designing programming languages, *Comm. ACM 16* (1973), 83-90.

[6] Fass,L.F., Learning context-free languages from their structured sentences, *SIGACT News*, Vol.15, No.3 (1983), 24-35.

[7] Gold,E.M., Complexity of automaton identification from given data, *Information and Control 10* (1967), 447-474.

[8] Gold,E.M., Language identification in the limit, *Information and Control 37* (1978), 302-320.

[9] Harrison,M.A., Introduction to formal language theory, Addison-Wesley, 1978.

[10] Hopcroft,J.E., Ullman,J.D., Introduction to automata theory, languages and computation, Addison-Wesley, 1979.

[11] Levy,L.S., Joshi,A.K., Skeletal structural descriptions, *Information and Control 39* (1978), 192-211.

[12] Paull,M.C., Unger,S.H., Structural equivalence of context-free grammars, *J. Comput. Sys. Sci. 2* (1968), 427-463.

[13] Takahashi,M., Generalizations of regular sets and their application to a study of context-free languages, *Information and Control 27* (1975), 1-36.

[14] Thatcher,J.W., Tree automaton : An informal survey, in *Currents in the Theory of Computing*, Aho,A.V., Ed., Prentice-Hall, 1973.

[15] Thatcher,J.W., Wright,J.B., Generalized finite automata theory with an application to a decision problem of second-order logic, *Mathem. Systems Theory 2* (1968), 57-81.