TR-488

An Efficient Learning of Context-Free Grammars from Positive Structural Examples

by

Y. Sakakibara (Fujitsu)

July, 1989

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

An Efficient Learning of Context-Free Grammars from Positive Structural Examples *

Yasubumi SAKAKIBARA

International Institute for Advanced Study of Social Information Science (IIAS-SIS)
FUJITSU LIMITED
140, Miyamoto, Numazu, Shizuoka, 410-03 Japan
E-mail : yasu%iias.fujitsu.co.jp@uunet.uu.net

* A preliminary version of the paper was presented at FGCS '88, ICOT, Tokyo, JAPAN.

Abstract

In this paper, we introduce a new normal form for context-free grammars, called *reversible context-free grammars*, for the problem of learning context-free grammars from positive-only examples. A context-free grammar $G = (N, \Sigma, P, S)$ is said to be *reversible* if (1) $A \rightarrow \alpha$ and $B \rightarrow \alpha$ in P implies $A = B$ and (2) $A \rightarrow \alpha B \beta$ and $A \rightarrow \alpha C \beta$ in P implies $B = C$. We show that the class of reversible context-free grammars is learnable from positive samples of structural descriptions and there exists an efficient algorithm to learn them from positive samples of structural descriptions, where a structural description of a context-free grammar is an unlabelled derivation tree of the grammar. This implies that if information on the structure of the grammar in the form of reversible is available to the learning algorithm, the full class of context-free languages can be learned efficiently from positive samples.

1 Introduction

We consider the problem of learning context-free languages from positive-only examples. The problem of learning a "correct" grammar for the unknown language from finite examples of the language is known as the grammatical inference problem. In the grammatical inference problem, however, there exists the computational hardness of it, and recently many researchers have turned their attention to the computational complexities of learning algorithms [5, 6, 8, 13, 14, 16, 17]. A criterion for evaluating the computational efficiency of a learning algorithm is the polynomial time bound, what is called *polynomial-time learnability*. Previously in order to solve the computational hardness of the inference problem of context-free grammars, we [16] have considered the problem of learning context-free grammars from their structural descriptions. A structural description of a context-free grammar is an unlabelled derivation tree of the grammar, that is, a derivation tree whose internal nodes have no label. Thus this problem setting assumes that information on the structure of the unknown grammar is available to the learning algorithm, which is also necessary to identify a grammar having the intended structure, that is, structurally equivalent to the unknown grammar. We showed an efficient algorithm to learn the full class of context-free grammars using two types of queries, structural membership and structural equivalence queries, in a teacher and learner paradigm which is introduced by Angluin [7] to model a learning situation in which a teacher is available to answer some queries about the material to be learned.

In Gold's criterion of identification in the limit for successful learning of a formal language, he [11] showed that there is a fundamental, important difference in what could be learned from positive versus complete samples. A positive sample presents all and only strings of the unknown language to the learning algorithm, while a complete sample presents all strings, each classified as to whether it belongs to the unknown language. Learning from positive samples is strictly weaker than learning from complete samples. Intuitively, an inherent difficulty in trying to learn from positive rather than complete samples depends on the problem of "overgeneralization". Gold showed that any class of languages containing all the finite languages and at least one infinite language cannot be identified in the limit from

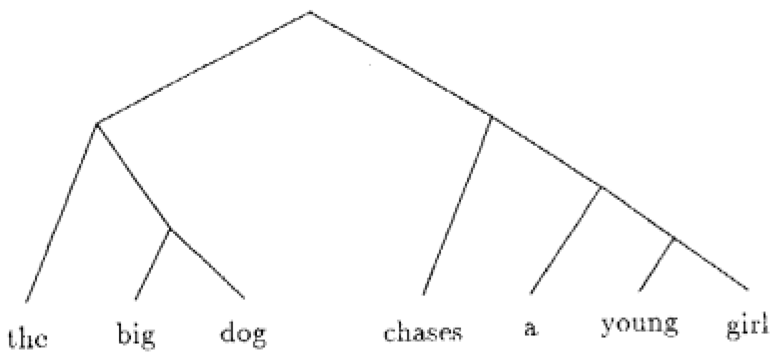

Figure 1: A structural description for "the big dog chases a young girl"

positive samples. According to this theoretical result, the class of context-free languages (even the class of regular sets) cannot be learned from positive samples. These facts seem to show that learning from positive samples is too weak to find practical and interesting applications of the grammatical inference. However it may be true that learning from positive samples is very useful and important for a practical use of the grammatical inference because it is very hard for the user to present and understand complete samples which force him to have a complete knowledge of the unknown (target) grammar.

In this paper, to overcome this essential difficulty of learning from positive samples, we again consider learning from structural descriptions, that is, we assume example presentations in the form of structural descriptions. The problem is to learn context-free grammars from positive samples of their structural descriptions, that is, all and only structural descriptions of the unknown grammar. We show that there is a class of context-free grammars, called *reversible context-free grammars*, which can be identified from positive samples of their structural descriptions and the reversible context-free grammar is a normal form for context-free grammars, that is, reversible context-free grammars can generate all of the context-free languages. We present a polynomial-time algorithm which identifies them in the limit from positive samples of their structural descriptions by extending Angluin's efficient algorithm [4] which identifies finite automata from positive samples to the one for tree automata. This implies that if information on the structure of the grammar in the form of reversible is available to the learning algorithm, the full class of context-free languages can be learned efficiently from positive samples.

We also demonstrate several examples to show the learning process of our learning algorithm and to emphasize how successfully and efficiently our learning algorithm identifies primary examples of grammars given in the previous works for the grammatical inference problem.

2 Basic Definitions

Let $\mathbf{N}$ be the set of positive integers and $\mathbf{N}^*$ be the free monoid generated by $\mathbf{N}$. For $y, x \in \mathbf{N}^*$, we write $y \leq x$ if and only if there is a $z \in \mathbf{N}^*$ such that $x = y \cdot z$, and $y < x$ if and only if $y \leq x$ and $y \neq x$.

A *ranked alphabet* V is a finite set of symbols associated with a finite relation called the *rank* relation $r_V \subseteq V \times \mathbf{N}$. V_n denotes the subset $\{f \in V \mid (f, n) \in r_V\}$ of V. Let $m = max\{n \mid V_n \neq \emptyset\}$, i.e., $m = min\{n \mid r_V \subseteq V \times \{0, 1, \ldots, n\}\}$. In many cases the symbols in V_n are considered as *function symbols*. We say that a function symbol f has an *arity* n if $f \in V_n$ and a symbol of arity 0 is called a *constant symbol*.

A *tree* over V is a mapping t from Dom_t into V where the domain Dom_t is a finite subset of $\mathbf{N}^*$ such that (1) if $x \in Dom_t$ and $y < x$, then $y \in Dom_t$; (2) if $y \cdot i \in Dom_t$ and $i \in \mathbf{N}$, then $y \cdot j \in Dom_t$ for $1 \leq j \leq i$, $j \in \mathbf{N}$; (3) $t(x) \in V_n$, whenever for $i \in \mathbf{N}$, $x \cdot i \in Dom_t$ if and only if $1 \leq i \leq n$. An element of the tree domain Dom_t is called a *node* of t. If $t(x) = A$, then we say that A is the *label* of the node x of t. V^T denotes the set of all trees over V. $|Dom_t|$ denotes the cardinality of Dom_t, that is, the number of nodes in t.

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

Let t be a tree over V. A node y in t is called a *terminal node* if and only if for all $x \in Dom_t$, $y \not< x$. A node y in t is an *internal node* if and only if y is not a terminal node. The *frontier* of Dom_t, denoted $frontier(Dom_t)$, is the set of all terminal nodes in Dom_t. The *interior* of Dom_t, denoted $interior(Dom_t)$, is $Dom_t - frontier(Dom_t)$. The *depth* of $x \in Dom_t$, denoted $depth(x)$, is the length of x. For a tree t, the *depth* of t is defined as $depth(t) = max\{depth(x) \mid x \in Dom_t\}$. The *size* of t is the number of nodes in t.

Let \$ be a new symbol (i.e., $\$ \notin V$) of rank 0. $V_\$^T$ denotes the set of all trees in $(V \cup \{\$\})^T$ which exactly contains one \$-symbol. For trees $s \in V_\$^T$ and $t \in (V^T \cup V_\$^T)$, we define an operation "#" to replace the terminal node labelled \$ of s with t by

$$s\#t(x) = \begin{cases} s(x) & \text{if } x \in Dom_s \text{ and } s(x) \neq \$, \\ t(y) & \text{if } x = z \cdot y,\ s(z) = \$ \text{ and } y \in Dom_t. \end{cases}$$

For subsets $S \subseteq V_\$^T$ and $T \subseteq (V^T \cup V_\$^T)$, $S\#T$ is defined to be the set $\{s\#t \mid s \in S \text{ and } t \in T\}$.

Let $t \in V^T$ and $x \in Dom_t$. The *subtree* t/x of t at x is a tree such that $Dom_{t/x} = \{y \mid x \cdot y \in Dom_t\}$ and $t/x(y) = t(x \cdot y)$ for any $y \in Dom_{t/x}$. The *co-subtree* $t\backslash x$ of t at x is a tree in $V_\$^T$ such that $Dom_{t\backslash x} = \{y \mid y \in Dom_t \text{ and } x \not< y\}$ and

$$t\backslash x(y) = \begin{cases} t(y) & \text{for } y \in Dom_{t\backslash x} - \{x\}, \\ \$ & \text{for } y = x. \end{cases}$$

Let T be a set of trees. We define the set $Sc(T)$ of co-subtrees of elements of T by

$$Sc(T) = \{t\backslash x \mid t \in T \text{ and } x \in Dom_t\},$$

and the set $Sub(T)$ of subtrees of elements of T by

$$Sub(T) = \{t/x \mid t \in T \text{ and } x \in Dom_t\}.$$

Also, for any $t \in V^T$, we denote the *quotient* of T and t by

$$U_T(t) = \begin{cases} \{u \mid u \in V_\$^T \text{ and } u\#t \in T\} & \text{if } t \in V^T - V_0, \\ t & \text{if } t \in V_0. \end{cases}$$

A *partition* of some set S is a set of pairwise disjoint nonempty subsets of S whose union is S. If π is a partition of S, then for any element $s \in S$ there is a unique element of π

containing s, which we denote $B(s,\pi)$ and call the *block* of π containing s. A partition π is said to *refine* another partition π', or π is *finer* than π', if and only if every block of π' is a union of blocks of π. If π is a partition of a set S and S' is a subset of S, then the *restriction* of π to S' is the partition π' consisting of all those sets E' that are nonempty and are the intersection of S' and some block of π. The *trivial partition* of a set S is the class of all singleton sets $\{s\}$ such that $s \in S$. An *algebraic congruence* is a partition π of V^T with the property that for $t_i, u_i \in V^T (1 \le i \le k)$ and $f \in V_k$, $B(t_i,\pi) = B(u_i,\pi)$ implies $B(f(t_1,\ldots,t_k),\pi) = B(f(u_1,\ldots,u_k),\pi)$. If T is any set of trees, then for $1 \le i \le k$ and $f \in V_k$, $U_T(t_i) = U_T(u_i)$ implies $U_T(f(t_1,\ldots,t_k)) = U_T(f(u_1,t_2,\ldots,t_k)) = \cdots = U_T(f(u_1,\ldots,u_{k-1},t_k)) = U_T(f(u_1,\ldots,u_k))$, so T determines an associated algebraic congruence π_T by $B(t_1,\pi_T) = B(t_2,\pi_T)$ if and only if $U_T(t_1) = U_T(t_2)$.

Definition Let V be a ranked alphabet and m be the maximum rank of the symbols in V. A *(frontier-to-root) tree automaton* over V is a quadruple $A = (Q, V, \delta, F)$ such that Q is a finite set, F is a subset of Q, and $\delta = (\delta_0, \delta_1, \ldots, \delta_m)$ consists of the following maps:

$$\delta_k : V_k \times (Q \cup V_0)^k \mapsto 2^Q \qquad (k = 1, 2, \ldots, m),$$

$$\delta_0(a) = a \qquad \text{for } a \in V_0.$$

Q is the set of *states*, F is the set of *final states* of A, and δ is the *state transition function* of A. In this definition, the terminal symbols on the frontier are taken as "initial" states. δ can be extended to V^T by letting :

$$\delta(f(t_1,\ldots,t_k)) = \begin{cases} \displaystyle\bigcup_{q_1\in\delta(t_1),\ldots,q_k\in\delta(t_k)} \delta_k(f,q_1,\ldots,q_k) & \text{if } k > 0, \\ \{f\} & \text{if } k = 0. \end{cases}$$

The tree t is *accepted* by A if and only if $\delta(t) \cap F \neq \emptyset$. The set of trees accepted by A, denoted $T(A)$, is defined as $T(A) = \{t \in V^T \mid \delta(t) \cap F \neq \emptyset\}$.

Note that the tree automaton A cannot accept any tree of depth 0.

A tree automaton is *deterministic* if and only if for each k-tuple $q_1,\ldots,q_k \in Q \cup V_0$ and each symbol $f \in V_k$, there is at most one element in $\delta_k(f,q_1,\ldots,q_k)$. Note that we allow undefined state transitions in deterministic tree automata.

Proposition 1 ([15]) *Nondeterministic tree automata are no more powerful than deterministic tree automata. That is, the set of trees accepted by a nondeterministic tree automaton is accepted by a deterministic tree automaton.*

Remark 1 *Let A be a deterministic tree automaton. If $\delta(t_1) = \delta(t_2)$, then $U_{T(A)}(t_1) = U_{T(A)}(t_2)$.*

Note that $\pi_{T(A)}$ contains finitely many blocks for any tree automaton A.

Let $A = (Q, V, \delta, F)$ and $A' = (Q', V, \delta', F')$ be tree automata. A is *isomorphic* to A' if and only if there exists a bijection φ of Q onto Q' such that $\varphi(F) = F'$ and for every $q_1, \ldots, q_k \in Q \cup V_0$ and $f \in V_k$, $\varphi(\delta_k(f, q_1, \ldots, q_k)) = \delta'_k(f, q'_1, \ldots, q'_k)$ where $q'_i = \varphi(q_i)$ if $q_i \in Q$ and $q'_i = q_i$ if $q_i \in V_0$ for $1 \leq i \leq k$.

Definition Let $A = (Q, V, \delta, F)$ and $A' = (Q', V, \delta', F')$ be tree automata. A' is a *tree subautomaton* of A if and only if Q' and F' are subsets of Q and F respectively and for every $q'_1, \ldots, q'_k \in Q' \cup V_0$ and $f \in V_k$, $\delta'_k(f, q'_1, \ldots, q'_k) = \delta_k(f, q'_1, \ldots, q'_k)$ or $\delta'_k(f, q'_1, \ldots, q'_k)$ is undefined.

Clearly $T(A') \subseteq T(A)$.

Definition Let $A = (Q, V, \delta, F)$ be a tree automaton. If Q'' is a subset of Q, then the *tree subautomaton of A induced by Q''* is the tree automaton (Q'', V, δ'', F''), where F'' is the intersection of Q'' and F, and $q'' \in \delta''_k(f, q''_1, \ldots, q''_k)$ if and only if $q'' \in Q''$, $q''_1, \ldots, q''_k \in Q'' \cup V_0$, and $q'' \in \delta_k(f, q''_1, \ldots, q''_k)$.

A state q of A is called *useful* if and only if there exist a tree t and some address $x \in Dom_t$ such that $\delta(t/x) = q$ and $\delta(t) \in F$. States that are not useful are called *useless*. A tree automaton that contains no useless states is called *stripped*.

Definition The *stripped tree subautomaton* of A is the tree subautomaton of A induced by the useful states of A.

Definition Let $A = (Q, V, \delta, F)$ be any tree automaton. If π is any partition of Q, we define another tree automaton $A/\pi = (Q', V, \delta', F')$ induced by π as follows: Q' is the set of blocks of π (i.e. $Q' = \pi$). F' is the set of all blocks of π that contain an element of F (i.e. $F' = \{B \in \pi \mid B \cap F \neq \emptyset\}$). δ' is a mapping from $V_k \times (\pi \cup V_0)^k$ to 2^π and for $B_1, \ldots, B_k \in Q' \cup V_0$ and $f \in V_k$, the block B is in $\delta'_k(f, B_1, \ldots, B_k)$ whenever there exist $q \in B$ and $q_i \in B_i \in \pi$ or $q_i = B_i \in V_0$ for $1 \leq i \leq k$ such that $q = \delta_k(f, q_1, \ldots, q_k)$.

Remark 2 *Let $A = (Q, V, \delta, F)$ be a tree automaton and π be a partition of Q. Then $T(A/\pi) \supseteq T(A)$, $T(A/\pi) = T(A)$ if π is the trivial partition of Q, and $T(A/\pi) \subseteq T(A/\pi')$ if π refines π'.*

Definition Let T be a set of trees accepted by some tree automaton. We define the *canonical tree automaton* for T, denoted $C(T) = (Q, V, \delta, F)$, as follows :

$$\begin{aligned} Q &= \{U_T(u) \mid u \in Sub(T) - V_0\}, \\ F &= \{U_T(t) \mid t \in T\}, \\ \delta_k(f, U_T(u_1), \ldots, U_T(u_k)) &= U_T(f(u_1, \ldots, u_k)) \\ &\qquad \text{if } u_1, \ldots, u_k \text{ and } f(u_1, \ldots, u_k) \text{ are in } Sub(T), \\ \delta_0(a) &= a \qquad \text{for } a \in V_0. \end{aligned}$$

Since T is accepted by some tree automaton, by Remark 1, the set $\{U_T(u) \mid u \in Sub(T) - V_0\}$ becomes finite. Since $U_T(u_1) = U_T(u_2)$ implies $U_T(t\#u_1) = U_T(t\#u_2)$ for all trees t in $V_\$^T$, this state transition function is well defined and $C(T)$ is deterministic. $C(T)$ is stripped, that is, contains no useless states. A tree automaton A is called *canonical* if and only if A is isomorphic to the canonical tree automaton for $T(A)$.

Definition Let Sa be a finite set of trees of V^T. We define the *base tree automaton* for Sa, denoted $Bs(Sa) = (Q, V, \delta, F)$, as follows :

$$\begin{aligned} Q &= Sub(Sa) - V_0, \\ F &= Sa, \end{aligned}$$

$$\delta_k(f, u_1, \ldots, u_k) = f(u_1, \ldots, u_k)$$

whenever $u_1, \ldots, u_k \in Q \cup V_0$ and $f(u_1, \ldots, u_k) \in Q$,

$$\delta_0(a) = a \qquad \text{for } a \in V_0.$$

Note that $Bs(Sa)$ is a tree automaton that accepts precisely the set Sa.

An *alphabet* is a finite non-empty set of symbols. The set of all finite strings of symbols in an alphabet Σ is denoted Σ^*. The empty string is denoted ϵ. The length of the string w is denoted $|w|$. If X is a finite set, $|X|$ denotes the cardinality of X.

Definition A *context-free grammar* is denoted $G = (N, \Sigma, P, S)$, where N and Σ are alphabets of *nonterminals* and *terminals* respectively such that $N \cap \Sigma = \emptyset$. P is a finite set of productions; each production is of the form $A \to \alpha$, 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*. If $A \to \beta$ is a production of P, then for any strings α and γ in $(N \cup \Sigma)^*$, we define $\alpha A \gamma \Rightarrow \alpha\beta\gamma$. $\stackrel{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$. The *language generated* by G, denoted $L(G)$, is $\{w \mid w \text{ is in } \Sigma^* \text{ and } S \stackrel{*}{\Rightarrow} w\}$.

Two context-free grammars G and G' are said to be *equivalent* if and only if $L(G) = L(G')$. Two context-free grammars $G = (N, \Sigma, P, S)$ and $G' = (N', \Sigma, P', S')$ are said to be *isomorphic*, that is, differ only by the names of nonterminals, if and only if there exists a bijection φ of N onto N' such that $\varphi(S) = S'$ and for every $A, B_1, \ldots, B_k \in N \cup \Sigma$, $A \to B_1 \cdots B_k \in P$ if and only if $\varphi(A) \to B'_1 \cdots B'_k \in P'$ where $B'_i = \varphi(B_i)$ if $B_i \in N$ and $B'_i = B_i$ if $B_i \in \Sigma$ for $1 \le i \le k$.

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 :

$$D_A(G) = \begin{cases} \{a\} & \text{if } A = a \in \Sigma, \\ \{A(t_1, \ldots, t_k) \mid A \to B_1 \cdots B_k,\ t_i \in D_{B_i}(G)\ (1 \le i \le k)\} & \text{if } A \in N. \end{cases}$$

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.

A *skeletal alphabet* Sk is a ranked alphabet consisting of only the special symbol σ with the rank relation $r_{Sk} \subseteq \{\sigma\} \times \{1, 2, 3, \ldots, m\}$, where m is the maximum rank of the symbols in the alphabet Sk. A tree defined over $Sk \cup V_0$ is called a *skeleton*.

Definition Let $t \in V^T$. The *skeletal* (or *structural*) *description* of t, denoted $s(t)$, is a skeleton with $Dom_{s(t)} = Dom_t$ such that

$$s(t)(x) = \begin{cases} t(x) & \text{if } x \in frontier(Dom_t), \\ \sigma & \text{if } x \in interior(Dom_t). \end{cases}$$

Let T be a set of trees. The *corresponding skeletal set*, denoted $K(T)$, is $\{s(t) \mid t \in T\}$.

Thus a skeleton is a tree defined over $Sk \cup \Sigma$ which has a special label σ for the internal nodes. The skeletal description of a tree preserves the structure of the tree, but not the label names describing that structure. A tree automaton over $Sk \cup \Sigma$ is called a *skeletal tree automaton*.

A skeleton in $K(D(G))$ is called a *structural description* of G. Then $K(D(G))$ is the set of structural descriptions of G. Two context-free grammars G and G' are said to be *structurally equivalent* if and only if $K(D(G)) = K(D(G'))$. Note that if G and G' are structurally equivalent, they are equivalent, too. Given a context-free grammar G, we can get the skeletal alphabet which $K(D(G))$ is defined over. Let r be the set of the lengths of the right-hand sides of all the productions in G. Then the skeletal alphabet Sk for $K(D(G))$ consists of $\{\sigma\}$ with $r_{Sk} = \{\sigma\} \times r$.

Next we show two important propositions which connect a context-free grammar with a tree automaton. By a coding of the derivation process of a context-free grammar in the formalism of a tree automaton, we can get the following result.

Definition Let $G = (N, \Sigma, P, S)$ be a context-free grammar. The corresponding skeletal tree automaton $A(G) = (Q, Sk \cup \Sigma, \delta, F)$ is defined as follows:

$Q = N,$

$F = \{S\},$

$\delta_k(\sigma, B_1, \ldots, B_k) \ni A$ if the production of the form $A \to B_1 \cdots B_k$ is in P,

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

Proposition 2 *Let G be a context-free grammar. Then $T(A(G)) = K(D(G))$. That is, the set of trees accepted by $A(G)$ is equal to the set of structural descriptions of G.*

Proof. First we prove that $s \in K(D_A(G))$ if and only if $\delta(s) \ni A$ 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 $A(G)$, $a \in D_A(G)$ if and only if $A = a$ if and only if $\delta(a) = \{\delta_0(a)\} \ni A$. Hence $a \in K(D_A(G))$ if and only if $\delta(a) \ni A$.

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 $s = \sigma(u_1, \ldots, u_k)$ for some skeletons $u_1, \ldots, u_k$ with depth at most h. Assume that $u_i \in K(D_{B_i}(G))$ for $1 \leq i \leq k$. Then

$\sigma(u_1, \ldots, u_k) \in K(D_A(G))$

if and only if there is the production of the form $A \to B_1 \cdots B_k$ in P,

by the definition of $D_A(G)$,

if and only if $\delta_k(\sigma, B_1, \ldots, B_k) \ni A$, by the definition of $A(G)$,

if and only if $\delta_k(\sigma, B_1, \ldots, B_k) \ni A$ and $B_1 \in \delta(u_1), \ldots, B_k \in \delta(u_k)$,

by the induction hypothesis,

if and only if $\delta(\sigma(u_1, \ldots, u_k)) \ni A$.

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

Then it immediately follows from this that $s \in K(D(G))$ if and only if $\delta(s) \ni S$. Hence $K(D(G)) = T(A(G))$. Q.E.D.

Conversely, by a coding of the recognizing process of a tree automaton in the formalism of a context-free grammar, we can get the following result.

Definition Let $A = (Q, Sk \cup \Sigma, \delta, F)$ be a deterministic skeletal tree automaton for a skeletal set. The corresponding context-free grammar $G(A) = (N, \Sigma, P, S)$ is defined as follows:

$$\begin{aligned} N &= Q \cup \{S\}, \\ P &= \{\delta_k(\sigma, x_1, \ldots, x_k) \to x_1 \cdots x_k \\ &\qquad \mid \sigma \in Sk_k,\ x_1, \ldots, x_k \in Q \cup \Sigma \text{ and } \delta_k(\sigma, x_1, \ldots, x_k) \text{ is defined}\} \\ &\quad \cup \{S \to x_1 \cdots x_k \mid \delta_k(\sigma, x_1, \ldots, x_k) \in F\}. \end{aligned}$$

Proposition 3 *Let $A = (Q, Sk \cup \Sigma, \delta, F)$ be a skeletal tree automaton. Then $K(D(G(A))) = T(A)$. That is, the set of structural descriptions of $G(A)$ is equal to the set of trees accepted by A.*

Proof. First we prove that (i) $\delta(s) = q$ if and only if $s \in K(D_q(G(A)))$ 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 of $G(A)$ and $D_A(G)$, $\delta(a) = q$ if and only if $q = a$ if and only if $a \in D_q(G(A))$. Hence $\delta(a) = q$ if and only if $a \in K(D_q(G(A)))$.

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 $s = \sigma(u_1, \ldots, u_k)$ for some skeletons $u_1, \ldots, u_k$ with depth at most h. Assume that $\delta(u_i) = x_i$ for $1 \le i \le k$. Then

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

if and only if $\delta_k(\sigma, \delta(u_1), \ldots, \delta(u_k)) = q$

if and only if $\delta_k(\sigma, x_1, \ldots, x_k) = q$

if and only if there is the production of the form $q \to x_1 \cdots x_k$ in $G(A)$,

by the definition of $G(A)$,

if and only if $q \to x_1 \cdots x_k$ in $G(A)$ and $u_1 \in K(D_{x_1}(G(A))), \ldots, u_k \in K(D_{x_k}(G(A)))$,

by the induction hypothesis,

if and only if $\sigma(u_1, \ldots, u_k) \in K(D_q(G(A)))$, by the definition of $D_A(G)$.

This completes the induction and the proof of (i).

Secondly we prove that (ii) $s \in K(D_S(G(A)))$ if and only if $s \in K(D_q(G(A)))$ for some $q \in F$. Let s be a skeleton of the form $\sigma(u_1, \ldots, u_k)$ for some skeletons $u_1, \ldots, u_k$. If $s \in K(D_S(G(A)))$, then since if $u_i \in K(D_{q_i}(G(A)))$, then $q_i = \delta(s_i)$ for $1 \le i \le k$ by (i), there is the production of the form $S \rightarrow \delta(u_1) \cdots \delta(u_k)$ in $G(A)$ and $\delta_k(\sigma, \delta(u_1), \ldots, \delta(u_k)) \in F$ by the definition of $G(A)$. Then $\delta(\sigma(u_1, \ldots, u_k)) \in F$ and so $\delta(s) \in F$. Hence by (i), $s \in K(D_q(G(A)))$ for some $q \in F$.

Conversely if $s \in K(D_q(G(A)))$ for some $q \in F$, then $\delta(s) = \delta_k(\sigma, \delta(u_1), \ldots, \delta(u_k)) \in F$ by (i). By the definition of $G(A)$, there is the production of the form $S \rightarrow \delta(u_1) \cdots \delta(u_k)$ in $G(A)$. Since $u_i \in K(D_{\delta(u_i)}(G(A)))$ for $1 \le i \le k$ by (i), $\delta(u_1, \ldots, u_k) \in K(D_S(G(A)))$. Hence $s \in K(D_S(G(A)))$.

Lastly it immediately follows from (i) and (ii) that $\delta(s) \in F$ if and only if $s \in K(D(G(A)))$. Hence $T(A) = K(D(G(A)))$. Q.E.D.

Therefore the problem of learning a context-free grammar from structural descriptions can be reduced to the problem of learning a tree automaton.

3 Structural Identification

Gold's theoretical study [11] of language learning introduces a fundamental concept that is very important in inductive inference : *identification in the limit*. In the Gold's traditional definition, to a learning algorithm M that is attempting to learn the unknown language L, an infinite sequence of examples of L is presented. A *positive presentation* of L is an infinite sequence giving all and only the elements of L. A *complete presentation* of L is an infinite sequence of ordered pairs (w, d) from $\Sigma^* \times \{0, 1\}$ such that $d = 1$ if and only if w is a member of L, and such that every element w of Σ^* appears as the first component of some pair in the sequence, where Σ is the alphabet which the language L is defined over. A positive presentation eventually includes every member of L, whereas a complete presentation eventually classifies every element of Σ^* as to its membership in L. If after some finite number of steps in a positive (complete) presentation of L, M guesses a correct conjecture for the unknown language L and never changes (*converges to*) its guess after this, then M is said to *identify L in the limit from positive (complete) samples*. In the case that the conjectures are in the form of grammars, M identifies in the limit a grammar G such that $L(G) = L$.

On the other hand, as indicated in [16], in order to identify a grammar which has the intended structure, it is necessary to assume that information on the structure of the grammar is available to the learning algorithm M. In the case of context-free grammars, the structure of a grammar is represented by the structural descriptions of it. Suppose G is the unknown context-free grammar (not the unknown language). This is the grammar that we assume has the intended structure, and that is to be learned (up to structural equivalence) by the learning algorithm M. In this case, a sequence of examples of the structural descriptions $K(D(G))$ is presented. A *positive presentation* of $K(D(G))$ is an infinite sequence giving all and only the elements of $K(D(G))$. A *complete presentation* of $K(D(G))$ is an infinite sequence of ordered pairs (s, d) from $(Sk \cup \Sigma)^* \times \{0, 1\}$ such that $d = 1$ if and only if s is a member of $K(D(G))$, and such that every element s of $(Sk \cup \Sigma)^*$ appears as the first component of some pair in the sequence, where Sk is the skeletal alphabet for the grammar G. Then a learning algorithm identifies in the limit a grammar G' such that $K(D(G')) = K(D(G))$ (i.e. structurally equivalent to G) from a presentation of the structural descriptions $K(D(G))$. This type of identification criterion is called *structural identification in the limit*.

4 Condition for Learning from Positive Samples

In order to learn formal languages from positive samples in the Gold's criterion of identification in the limit, we must avoid the problem of "overgeneralization", which means guessing a language that is a strict superset of the unknown language. Angluin showed in [3] various conditions for correct identification of formal languages from positive samples that avoids overgeneralization. In her framework, the target domain is an indexed family of nonempty recursive languages $L_1, L_2, L_3, \ldots$.

An indexed family of nonempty recursive languages $L_1, L_2, L_3, \ldots$ is said to be *learnable* *from positive (complete) samples* if and only if there exists a learning algorithm M which identifies L_i in the limit from positive (complete) samples for all $i \geq 1$.

One of necessary and sufficient conditions for correct identification from positive samples is following.

Condition 1 An indexed family of nonempty recursive languages $L_1, L_2, L_3, \ldots$ *satisfies Condition 1* if and only if there exists an effective procedure which on any input $i \geq 1$ enumerates a set of strings T_i such that

1. T_i is finite,
2. $T_i \subseteq L_i$, and
3. for all $j \geq 1$, if $T_i \subseteq L_j$ then L_j is not a proper subset of L_i.

This condition requires that for every language L_i, there exists a "telltale" finite subset T_i of L_i such that no language of the family that also contains T_i is a proper subset of L_i. Angluin proved that an indexed family of nonempty recursive languages is learnable from positive samples if and only if it satisfies Condition 1.

These characterizations and results can be easily applied to the problem of learning tree automata, and hence to the problem of structural identification of context-free grammars because the Angluin's results assume only the enumerability and recursiveness of a class of languages.

5 Reversible Context-Free Grammars

Definition A skeletal tree automaton $A = (Q, Sk \cup \Sigma, \delta, F)$ is *reset-free* if and only if for no two distinct states q_1 and q_2 in Q do there exist a symbol $\sigma \in Sk_k$, a state $q_3 \in Q$, an integer $i \in \mathbb{N}$ $(1 \leq i \leq k)$ and $k-1$-tuple $u_1, \ldots, u_{k-1} \in Q \cup \Sigma$ such that $\delta_k(\sigma, u_1, \ldots, u_{i-1}, q_1, u_i, \ldots, u_{k-1}) = q_3 = \delta_k(\sigma, u_1, \ldots, u_{i-1}, q_2, u_i, \ldots, u_{k-1})$. The skeletal tree automaton is said to be *reversible* if and only if it is deterministic, has at most one final state, and is reset-free.

The idea of the reversible skeletal tree automaton comes from the "reversible automaton" and the "reversible languages" in [4]. Basically, the reversible skeletal tree automaton is the extension of the "zero-reversible automaton".

Remark 3 *If A is a reversible skeletal tree automaton and A' is any tree subautomaton of A, then A' is a reversible skeletal tree automaton.*

Lemma 4 *Let $A = (Q, Sk \cup \Sigma, \delta, \{q_f\})$ be a reversible skeletal tree automaton. For $t \in (Sk \cup \Sigma)_\$^T$ and $u_1, u_2 \in (Sk \cup \Sigma)^T$, if A accepts both $t\#u_1$ and $t\#u_2$, then $\delta(u_1) = \delta(u_2)$.*

Proof. We prove it by induction on the depth of the node labelled \$ in t. Suppose first that $t = \$$. Since A has only one final state q_f, $\delta(u_1) = \delta(t\#u_1) = q_f = \delta(t\#u_2) = \delta(u_2)$. Next suppose that the result holds for all $t \in (Sk \cup \Sigma)_\$^T$ in which the depth of the node labelled \$ is at most h. Let t be an element of $(Sk \cup \Sigma)_\$^T$ in which the depth of the node labelled \$ is $h+1$, so that $t = t'\#\sigma(s_1, \ldots, s_{i-1}, \$, s_i, \ldots, s_{k-1})$ for some $s_1, \ldots, s_{k-1} \in (Sk \cup \Sigma)^T$, $i \in \mathbb{N}$ and $t' \in (Sk \cup \Sigma)_\$^T$ in which the depth of the node labelled \$ is h. If A accepts both $t\#u_1 = t'\#\sigma(s_1, \ldots, s_{i-1}, u_1, s_i, \ldots, s_{k-1})$ and $t\#u_2 = t'\#\sigma(s_1, \ldots, s_{i-1}, u_2, s_i, \ldots, s_{k-1})$, then $\delta(\sigma(s_1, \ldots, s_{i-1}, u_1, s_i, \ldots, s_{k-1})) = \delta(\sigma(s_1, \ldots, s_{i-1}, u_2, s_i, \ldots, s_{k-1}))$ by the induction hypothesis. So

$$\begin{aligned}
&\delta_k(\sigma, \delta(s_1), \ldots, \delta(s_{i-1}), \delta(u_1), \delta(s_i), \ldots, \delta(s_{k-1})) \\
= \; &\delta_k(\sigma, \delta(s_1), \ldots, \delta(s_{i-1}), \delta(u_2), \delta(s_i), \ldots, \delta(s_{k-1})).
\end{aligned}$$

Since A is reset-free, $\delta(u_1) = \delta(u_2)$, which completes the induction and the proof of Lemma 4. Q.E.D.

Definition A context-free grammar $G = (N, \Sigma, P, S)$ is said to be *invertible* if and only if $A \to \alpha$ and $B \to \alpha$ in P implies $A = B$.

The motivation for studying invertible grammars comes from the theory of bottom-up parsing. Bottom-up parsing consists of (1) successively finding phrases and (2) reducing them to their parents. In a certain sense, each half of this process can be made simple but only at the expense of the other. Invertible grammars allow reduction decisions to be made simply. Invertible grammars have unique righthand sides of the productions so that the reduction phase of parsing becomes a matter of table lookup. The invertible grammar is one of normal forms for context-free grammars. Thus for any context-free language L, there is an invertible grammar G such that $L(G) = L$.

Definition A context-free grammar $G = (N, \Sigma, P, S)$ is *reset-free* if and only if for any two nonterminals B, C and $\alpha, \beta \in (N \cup \Sigma)^*$, $A \rightarrow \alpha B \beta$ and $A \rightarrow \alpha C \beta$ in P implies $B = C$.

Definition A context-free grammar G is said to be *reversible* if and only if G is invertible and reset-free. A context-free language L is defined to be *reversible* if and only if there exists a reversible context-free grammar G such that $L = L(G)$.

Definition Let $A = (Q, Sk \cup \Sigma, \delta, \{q_f\})$ be a reversible skeletal tree automaton for a skeletal set. The corresponding context-free grammar $G'(A) = (N, \Sigma, P, S)$ is defined as follows.

$$
\begin{aligned}
N &= Q, \\
S &= q_f, \\
P &= \{\delta_k(\sigma, x_1, \ldots, x_k) \rightarrow x_1 \cdots x_k \\
&\qquad\qquad \mid \sigma \in Sk_k,\ x_1, \ldots, x_k \in Q \cup \Sigma \text{ and } \delta_k(\sigma, x_1, \ldots, x_k) \text{ is defined}\}.
\end{aligned}
$$

By the definitions of $A(G)$ and $G'(A)$, we can conclude the following.

Proposition 5 *If G is a reversible context-free grammar, then $A(G)$ is a reversible skeletal tree automaton such that $T(A(G)) = K(D(G))$. Conversely if A is a reversible skeletal tree automaton, then $G'(A)$ is a reversible context-free grammar such that $K(D(G'(A))) = T(A)$.*

Therefore the problem of structural identification of reversible context-free grammars is reduced to the problem of identification of reversible skeletal tree automata.

Next we show some important theorems about the normal form property of reversible context-free grammars. First we show that each context-free language can be given a reversible context-free grammar.

Theorem 6 *For any context-free language L, there is a reversible context-free grammar G such that $L(G) = L$.*

Proof. First we assume that L does not contain the empty string. Let $G' = (N', \Sigma, P', S')$ be a ϵ-free context-free grammar in Chomsky normal form such that $L(G') = L$. Index the productions in P' by the integers $1, 2, \ldots, |P'|$. Let the index of $A \rightarrow \alpha \in P'$ be denoted $I(A \rightarrow \alpha)$. Let R be a new nonterminal symbol not in N' and construct $G = (N, \Sigma, P, S)$ as follows:

$$
\begin{aligned}
N &= N' \cup \{R\}, \\
S &= S', \\
P &= \{A \rightarrow \alpha R^i \mid A \rightarrow \alpha \in P' \text{ and } i = I(A \rightarrow \alpha)\} \\
&\quad \cup \{R \rightarrow \epsilon\}
\end{aligned}
$$

Clearly G is reversible and $L(G) = L$.

If $\epsilon \in L$, let $L' = L - \{\epsilon\}$ and $G' = (N, \Sigma, P, S)$ be the reversible context-free grammar constructed in the above way for L'. Then $G = (N, \Sigma, P \cup \{S \rightarrow R\}, S)$ is reversible and $L(G) = L$. Q.E.D.

The trivialization occurs in the previous proof because ϵ-productions are used to encode the index of the production. We prefer to allow ϵ-production only if absolutely necessary and prefer ϵ-free reversible context-free grammars if possible. Unfortunately there are context-free languages for which there do not exist any ϵ-free reversible context-free grammar. An example of such a language is:

$$\{a^i \mid i \geq 1\} \cup \{b^j \mid j \geq 1\} \cup \{c\}$$

However if a context-free language does not contain the empty string and any terminal string of length one, then there is a ϵ-free reversible context-free grammar which generates the language. In order to obtain this useful result, we quote an important theorem for invertible grammars in [12].

Proposition 7 ([12]) *For each context-free grammar G there is an invertible context-free grammar G' so that $L(G') = L(G)$. Moreover, if G is ϵ-free then so is G'.*

Now we have the following. The argument of the proof becomes more complex but the result is more useful.

Theorem 8 *Let L be any context-free language in which all strings are of length at least two. Then there is a ϵ-free reversible context-free grammar G such that $L(G) = L$.*

Proof. We construct the reversible context-free grammar $G = (N, \Sigma, P, S)$ in the following steps.

First by the proof of the above proposition in [12], there is an invertible context-free grammar $G' = (N', \Sigma, P', S')$ such that $L(G') = L$ and each production in P' is of the form

1. $A \to BC$ with $A, B, C \in N' - \{S'\}$ or

2. $A \to a$ with $A \in N' - \{S'\}$ and $a \in \Sigma$ or

3. $S' \to A$ with $A \in N' - \{S'\}$.

Since all strings in L are of length at least two, P' has no production of the form $A \to a$ for $A \in N' - \{S'\}$ and $a \in \Sigma$ such that $S' \to A \in P'$.

Next for all productions in P' whose left-hand side is not the start symbol, we make them reset-free with preserving invertible. P'' is defined as follows:

1. For each $A \in N' - \{S'\}$, let

$$\{A \to \alpha_1, A \to \alpha_2, \ldots, A \to \alpha_n\}$$

   be the set of all productions in P' whose left-hand side is A. P'' contains the set of productions

$$\{A \to \alpha_1, A \to X_{A_1}, X_{A_1} \to \alpha_2, X_{A_1} \to X_{A_2}, \ldots, X_{A_{n-1}} \to \alpha_n\},$$

   where $X_{A_1}, X_{A_2}, \ldots, X_{A_{n-1}}$ are new distinct nonterminal symbols.

2. For each $A \in N' - \{S'\}$ such that $S' \to A \in P'$, let

$$\{A \to B_1C_1, A \to B_2C_2, \ldots, A \to B_nC_n\}$$

be the set of all productions in P' whose left-hand side is A. P'' contains the set of productions

$$\{S'' \to B_1\bar{C}_1, S'' \to B_2\bar{C}_2, \ldots, S'' \to B_n\bar{C}_n\},$$

where $\bar{C}_j$ $(1 \leq j \leq n)$ is a new nonterminal symbol.

3. P'' contains the set of productions $\{\bar{C} \to C \mid C \in N' - \{S'\}\}$.

Let $G'' = (N'', \Sigma, P'', S'')$, where $N'' = N' \cup \{X_{A_1}, X_{A_2}, \ldots, X_{A_{n-1}} \mid A \in N' - \{S'\}\} \cup \{\bar{C} \mid C \in N' - \{S'\}\} \cup \{S''\}$. Then it is obvious that G'' is invertible and $L(G'') = L(G')$.

Lastly for all productions in P'' whose left-hand side is the start symbol, we make them reset-free with preserving invertible and get the desired grammar. P is defined as follows:

1. $A \to \alpha \in P$ if $A \to \alpha \in P''$ and $A \neq S''$.

2. Let

$$\{S'' \to \alpha_1, S'' \to \alpha_2, \ldots, S'' \to \alpha_n\}$$

be the set of all productions in P'' whose left-hand side is S''. P contains the set of productions

$$\{S \to \alpha_1, S \to X_{S_1}, X_{S_1} \to \alpha_2, X_{S_1} \to X_{S_2}, \ldots, X_{S_{n-1}} \to \alpha_n\},$$

where $X_{S_1}, X_{S_2}, \ldots, X_{S_{n-1}}$ are new distinct nonterminal symbols.

Let $G = (N, \Sigma, P, S)$, where $N = N'' \cup \{X_{S_1}, X_{S_2}, \ldots, X_{S_{n-1}}\} \cup \{S\}$. Clearly the resulting grammar G is reversible, ϵ-free and $L(G) = L(G'')$, that is, G generates L. Q.E.D.

Definition A context-free grammar $G = (N, \Sigma, P, S)$ is said to be *extended reversible* if and only if for $P' = P - \{S \to a \mid a \in \Sigma\}$, $G' = (N, \Sigma, P', S)$ is reversible.

By the above theorem, reversible context-free grammars can be easily extended so that for any context-free language not containing ϵ, we can find an extended reversible context-free grammar which is ϵ-free and generates the language.

Theorem 9 *Let L be any context-free language not containing ϵ. Then there is a ϵ-free extended reversible context-free grammar G such that $L(G) = L$.*

Proof. It is obvious from the definition of the extended reversible context-free grammars and Theorem 8. Q.E.D.

6 Learning Algorithms

In this section we first describe and analyze the algorithm RT to learn reversible skeletal tree automata from positive samples. Next we apply this algorithm to learning context-free grammars from positive samples of their structural descriptions. Essentially the algorithm RT is an extension of Angluin's learning algorithm for zero-reversible automata [4]. Without loss of generality, we restrict our consideration to only ϵ-free context-free grammars.

Definition A *positive sample* of a tree automaton A is a finite subset of $T(A)$. A positive sample CS of a reversible skeletal tree automaton A is a *characteristic sample* for A if and only if for any reversible skeletal tree automaton A', $T(A') \supseteq CS$ implies $T(A) \subseteq T(A')$.

6.1 The Learning Algorithm RT for Tree Automata

The input to RT is a finite nonempty set of skeletons Sa. The output is a particular reversible skeletal tree automaton $A = RT(Sa)$. The learning algorithm RT begins with the base tree automaton for Sa and generalizes it by merging states. RT finds a reversible skeletal tree automaton whose characteristic sample is precisely the input sample.

On input Sa, RT first constructs $A = Bs(Sa)$, the base tree automaton for Sa. It then constructs the finest partition π_f of the set Q of states of A with the property that A/π_f is reversible, and outputs A/π_f.

To construct π_f, RT begins with the trivial partition of Q and repeatedly merges any two distinct blocks B_1 and B_2 if either of the following conditions is satisfied.

1. B_1 and B_2 both contain final states of A.

2. There exist two states $q \in B_1$ and $q' \in B_2$ of the forms $q = \sigma(u_1, \ldots, u_k)$ and $q' = \sigma(u'_1, \ldots, u'_k)$ such that for $1 \le j \le k$, u_j and u'_j both are in the same block or the same terminal symbols.

3. There exist two states q, q' of the forms $q = \sigma(u_1, \ldots, u_k)$ and $q' = \sigma(u'_1, \ldots, u'_k)$ in the same block and an integer l $(1 \le l \le k)$ such that $u_l \in B_1$ and $u'_l \in B_2$ and for $1 \le j \le k$ and $j \ne l$, u_j and u'_j both are in the same block or the same terminal symbols.

When there no longer remains any such pair of blocks, the resulting partition is π_f.

To implement this merging process, RT keeps track of the further merges immediately implied by each merge performed. The variable LIST contains a list of pairs of states whose corresponding blocks are to be merged. RT initially selects some final state q of A and places on LIST all pairs (q, q') such that q' is a final state of A other than q. This ensures that all blocks containing a final state of A will eventually be merged.

After these initializations, RT proceeds as follows. While the list LIST is nonempty, RT removes the first pair of states (q_1, q_2). If q_1 and q_2 are already in the same block of the current partition, RT goes on to the next pair of states in LIST. Otherwise, the blocks containing q_1 and q_2, call them B_1 and B_2, are merged to form a new block B_3. This action entails that LIST be updates as follows. For any two states $q, q' \in Q$ of the forms $q = \sigma(u_1, \ldots, u_k)$ and $q' = \sigma(u'_1, \ldots, u'_k)$, if q and q' are not in the same block and u_j and u'_j both are in the same block or the same terminal symbols for $1 \le j \le k$, then the pair (q, q') is added to LIST. Also for any $q \in B_1, q' \in B_2$ of the forms $q = \sigma(u_1, \ldots, u_k)$ and $q' = \sigma(u'_1, \ldots, u'_k)$ and an integer l $(1 \le l \le k)$, if u_l and u'_l are states of A and not in the same block and u_j and u'_j both are in the same block or the same terminal symbols for $1 \le j \le k$ and $j \ne l$, then the pair (u_l, u'_l) is added to LIST. After this updating, RT goes on to the next pair of states from LIST.

When LIST becomes empty, the current partition is π_f. RT outputs A/π_f and halts.

The learning algorithm RT is illustrated in Figure 2. This completes the description of the algorithm RT, and we next analyze its correctness.

Input : a nonempty positive sample Sa;
Output : a reversible skeletal tree automaton A;
Procedure :
%% Initialization
Let $A = (Q, V, \delta, F)$ be $Bs(Sa)$;
Let π_0 be the trivial partition of Q;
Choose some $q \in F$;
Let LIST contain all pairs (q, q') such that $q' \in F - \{q\}$;
Let $i = 0$;
%% Main Routine
%% Merging
While LIST$\neq \emptyset$ **do**
    Begin
        Remove first element (q_1, q_2) from LIST;
        Let $B_1 = B(q_1, \pi_i)$ and $B_2 = B(q_2, \pi_i)$;
        If $B_1 \neq B_2$ **then**
          Begin
              Let π_{i+1} be π_i with B_1 and B_2 merged;
              p-UPDATE(π_{i+1}) and s-UPDATE(π_{i+1}, B_1, B_2);
              Increase i by 1;
          End
    End
%% Termination
Let $f = i$ and output the tree automaton A/π_f.
%% Sub-routine
where
    p-UPDATE(π_{i+1}) is :
      For all pairs of states $\sigma(u_1, \ldots, u_k)$ and $\sigma(u'_1, \ldots, u'_k)$ in Q with
          $B(u_j, \pi_{i+1}) = B(u'_j, \pi_{i+1})$ or $u_j = u'_j \in \Sigma$ for $1 \leq j \leq k$
          and $B(\sigma(u_1, \ldots, u_k), \pi_{i+1}) \neq B(\sigma(u'_1, \ldots, u'_k), \pi_{i+1})$
      do
        Add the pair $(\sigma(u_1, \ldots, u_k), \sigma(u'_1, \ldots, u'_k))$ to LIST;
    s-UPDATE(π_{i+1}, B_1, B_2) is :
      For all pairs of states $\sigma(u_1, \ldots, u_k) \in B_1$ and $\sigma(u'_1, \ldots, u'_k) \in B_2$ with
          $u_l, u'_l \in Q$ and $B(u_l, \pi_{i+1}) \neq B(u'_l, \pi_{i+1})$ for some l $(1 \leq l \leq k)$
          and $B(u_j, \pi_{i+1}) = B(u'_j, \pi_{i+1})$ or $u_j = u'_j \in \Sigma$ for $1 \leq j \leq k$ and $j \neq l$
      do
        Add the pair (u_l, u'_l) to LIST.

Figure 2: The learning algorithm RT for Reversible Tree Automata

6.2 Correctness of RT

In this section, we show that RT correctly finds a reversible skeletal tree automaton whose characteristic sample is precisely the input sample.

Lemma 10 *Let Sa be a positive sample of some tree automaton A. Let π be the partition $\pi_{T(A)}$ restricted to the set $Sub(Sa)-\Sigma$. Then $Bs(Sa)/\pi$ is isomorphic to a tree subautomaton of the canonical tree automaton $C(T(A))$. Furthermore, $T(Bs(Sa)/\pi)$ is contained in $T(A)$.*

Proof. The result holds trivially if $Sa = \emptyset$, so assume that $Sa \neq \emptyset$. Let $Bs(Sa)/\pi = (Q, V, \delta, F)$ and $C(T(A)) = (Q', V, \delta', F')$. The partition π is defined by $B(t_1, \pi) = B(t_2, \pi)$ if and only if $U_{T(A)}(t_1) = U_{T(A)}(t_2)$, for all $t_1, t_2 \in Sub(Sa) - \Sigma$. Hence $h(B(t, \pi)) = U_{T(A)}(t)$ is a well-defined and injective map from Q to Q'. If B_1 is a final state of $Bs(Sa)/\pi$, then $B_1 = B(t, \pi)$ for some t in Sa, and since $T(A)$ contains Sa, $U_{T(A)}(t)$ is a final state of $C(T(A))$. Hence h maps F to F'.

$Bs(Sa)/\pi$ is deterministic because for $f(t_1, \ldots, t_k)$ and $f(u_1, \ldots, u_k)$ in $Sub(Sa)$, $B(t_i, \pi) = B(u_i, \pi)$ if $t_i, u_i \in Sub(Sa)-\Sigma$ and $t_i = u_i$ if $t_i, u_i \in \Sigma$ $(1 \leq i \leq k)$ imply $B(f(t_1, \ldots, t_k), \pi) = B(f(u_1, \ldots, u_k), \pi)$. For $q_1, \ldots, q_k \in Q \cup \Sigma$ and $f \in V_k$,

$$\begin{aligned} h(\delta_k(f, q_1, \ldots, q_k)) &= h(B(f(t_1, \ldots, t_k), \pi)), \\ &\qquad \text{where } B(t_i, \pi) = q_i \text{ if } q_i \in Q \text{ and } t_i = q_i \text{ if } q_i \in \Sigma \ (1 \leq i \leq k), \\ &= U_{T(A)}(f(t_1, \ldots, t_k)) \\ &= \delta'_k(f, U_{T(A)}(t_1), \ldots, U_{T(A)}(t_k)). \end{aligned}$$

Thus h is an isomorphism between $Bs(Sa)/\pi$ and a tree subautomaton of $C(T(A))$. Q.E.D.

Lemma 11 *Suppose A is a reversible skeletal tree automaton. Then the stripped tree subautomaton A' of A is canonical.*

Proof. By Remark 3, A' is a reversible skeletal tree automaton, and accepts $T = T(A)$. If $T = \emptyset$, then A' is the tree automaton with the empty set of states and therefore canonical.

So suppose that $T \neq \emptyset$. Let $C(T) = (Q, Sk \cup \Sigma, \delta, \{q_f\})$ and $A' = (Q', Sk \cup \Sigma, \delta', \{q'_f\})$. We define $h(q') = U_T(u)$ if $\delta'(u) = q'$ for $q' \in Q'$. By Remark 1, h is a well-defined and surjective map from Q' to Q. Let q'_1 and q'_2 be states of A', and suppose that $U_T(u_1) = U_T(u_2)$ for u_1 and u_2 such that $\delta'(u_1) = q'_1$ and $\delta'(u_2) = q'_2$. Since A' is stripped, this implies that there exists a tree $t \in (Sk \cup \Sigma)^T_\$$ such that $t\#u_1$ and $t\#u_2$ are in T. Thus, by Lemma 4, $q'_1 = q'_2$. Hence h is injective. Since $\delta'(u) = q'_f$ for any $u \in T$, h maps $\{q'_f\}$ to $\{q_f\}$. For $q'_1, \ldots, q'_k \in Q' \cup \Sigma$ and $\sigma \in Sk_k$,

$$\begin{aligned} h(\delta'_k(\sigma, q'_1, \ldots, q'_k)) &= h(\delta'(\sigma(u_1, \ldots, u_k))), \\ &\qquad \text{where } \delta'(u_i) = q'_i \text{ for } 1 \leq i \leq k, \\ &= U_T(\sigma(u_1, \ldots, u_k)) \\ &= \delta_k(\sigma, U_T(u_1), \ldots, U_T(u_k)). \end{aligned}$$

Thus h is an isomorphism between $C(T)$ and A'. Hence A' is canonical. Q.E.D.

Lemma 12 *Suppose that A is a reversible skeletal tree automaton. Then the canonical tree automaton $C(T(A))$ is reversible.*

Proof. By the above lemma and Remark 3, the stripped tree subautomaton A' of A is canonical, reversible, and accepts $T(A)$. Thus, since $C(T(A))$ is isomorphic to A', $C(T(A))$ is reversible. Q.E.D.

Lemma 13 *Let Sa be any nonempty positive sample of skeletons, and π_f be the final partition found by RT on input Sa. Then π_f is the finest partition such that $Bs(Sa)/\pi_f$ is reversible.*

Proof. Let $A = (Q, Sk \cup \Sigma, \delta, F)$ be $Bs(Sa)$. If the pair (q_1, q_2) is ever placed on LIST, then q_1 and q_2 must be in the same block of the final partition, that is, $B(q_1, \pi_f) = B(q_2, \pi_f)$. Therefore, the initialization guarantees that all the final states of A are in the same block of π_f, so A/π_f has exactly one final state. For any $B_1, \ldots, B_k \in \pi_f \cup \Sigma$ and $\sigma \in Sk_k$, all the elements of $\delta_k(\sigma, B_1, \ldots, B_k)$ are contained in one block of π_f. Thus A/π_f is deterministic. Also, for any block B of π_f, any pair of states $q_1, q_2 \in B$ of the forms $q_1 = \sigma(u_1, \ldots, u_k)$ and

$q_2 = \sigma(u'_1, \ldots, u'_k)$ and any integer l ($1 \le l \le k$), if $B(u_j, \pi_f) = B(u'_j, \pi_f)$ or $u_j = u'_j \in \Sigma$ for $1 \le j \le k$ and $j \ne l$, then both u_l and u'_l are in the same block or the same terminal symbols. Thus A/π_f is reset-free. Hence A/π_f is reversible.

Next we show that if π is any partition of Q such that A/π is reversible, then π_f refines π. We prove by induction that π_i refines π for $i = 0, 1, \ldots, f$. Clearly π_0, the trivial partition of Q, refines π. Suppose that $\pi_0, \pi_1, \ldots, \pi_i$ all refines π and π_{i+1} is obtained from π_i by merging the blocks $B(q_1, \pi_i)$ and $B(q_2, \pi_i)$ in the course of processing entry (q_1, q_2) from LIST. Since π_i refines π, $B(q_1, \pi_i)$ is a subset of $B(q_1, \pi)$ and $B(q_2, \pi_i)$ is a subset of $B(q_2, \pi)$. So in order to show that π_{i+1} refines π, it is sufficient to show that $B(q_1, \pi) = B(q_2, \pi)$.

If (q_1, q_2) was first placed on LIST during the initialization stage, then q_1 and q_2 are both final states, and since A/π is reversible, it has only one final state, and so $B(q_1, \pi) = B(q_2, \pi)$. Otherwise, (q_1, q_2) was first placed on LIST in consequence of some previous merge, say the merge to produce π_m from π_{m-1}, where $0 < m \le i$. Then either q_1 and q_2 are of the forms $\sigma(u_1, \ldots, u_k)$ and $\sigma(u'_1, \ldots, u'_k)$ respectively and $B(u_j, \pi_m) = B(u'_j, \pi_m)$ or $u_j = u'_j \in \Sigma$ for $1 \le j \le k$, or there exist two states q'_1 in the block B_1 and q'_2 in the block B_2 of the forms $\sigma(u_1, \ldots, u_{l-1}, q_1, u_l, \ldots, u_{k-1})$ and $\sigma(u'_1, \ldots, u'_{l-1}, q_2, u'_l, \ldots, u'_{k-1})$ respectively for some l ($1 \le l \le k$) such that $B(u_j, \pi_m) = B(u'_j, \pi_m)$ or $u_j = u'_j \in \Sigma$ for $1 \le j \le k-1$, where B_1 and B_2 are the blocks of π_{m-1} merged in forming π_m. Since π_m refines π by the induction hypothesis and A/π is reversible, $B(q_1, \pi) = B(q_2, \pi)$. Thus in either case π_{i+1} refines π. Hence by finite induction we conclude that π_f refines π. Q.E.D.

Theorem 14 *Let Sa be a nonempty positive sample of skeletons, and A_f be the skeletal tree automaton output by the algorithm RT on input Sa. Then for any reversible skeletal tree automaton A, $T(A) \supseteq Sa$ implies $T(A_f) \subseteq T(A)$.*

Proof. The preceding lemma shows that A_f is a reversible skeletal tree automaton such that $T(A_f) \supseteq Sa$. Let A be any reversible skeletal tree automaton such that $T(A) \supseteq Sa$, and π be the restriction of the partition $\pi_{T(A)}$ to the set $Sub(Sa) - \Sigma$. Lemma 10 shows that $Bs(Sa)/\pi$ is isomorphic to a tree subautomaton of $C(T(A))$ and $T(Bs(Sa)/\pi)$ is contained in $T(A)$. Lemma 12 shows that $C(T(A))$ is reversible, and therefore by Remark 3, $Bs(Sa)/\pi$

is reversible. Let π_f be the final partition found by RT. By the above lemma, π_f refines π, so $T(Bs(Sa)/\pi_f) = T(A_f)$ is contained in $T(Bs(Sa)/\pi)$ by Remark 2. Hence, $T(A_f)$ is contained in $T(A)$. Q.E.D.

6.3 Time Complexity of RT

Theorem 15 *The algorithm RT may be implemented to run in time polynomial in the sum of the sizes of the input skeletons, where the size of a skeleton (or tree) t is the number of nodes in t, i.e. $|Dom_t|$.*

Proof. Let Sa be the set of input skeletons, n be the sum of the sizes of the skeletons in Sa, and d be the maximum rank of the symbol σ in Sk. The base tree automaton $A = Bs(Sa)$ may be constructed in time $O(n)$ and contains at most n states. Similarly, the time to output the final tree automaton is $O(n)$. The partitions π_i of the states of A may be queried and updated using the simple MERGE and FIND operations described in [1]. Processing each pair of states from LIST entails two FIND operations to determine the blocks containing the two states. If the blocks are distinct, which can happen at most $n-1$ times, they are merged with a MERGE operation, and p-UPDATE and s-UPDATE procedures process $2(d+1)n(n-1)$ and at most $2dn(n-1)$ FIND operations respectively. Further at most $n-1$ new pairs may be placed on LIST. Thus a total of at most $2n(n-1)+(n-1)$ pairs must be placed on LIST. Thus at most $2((2d+1)n(n-1)+2n+1)(n-1)$ FIND operations and $n-1$ MERGE operations are required. The operation MERGE takes $O(n)$ time and the operation FIND takes constant time, so RT requires a total time of $O(n^3)$. Q.E.D.

6.4 Identification in the Limit of Reversible Tree Automata

Next we show that the algorithm RT may be used at the finite stages of an infinite learning process to identify the reversible skeletal tree automata in the limit from positive samples. The idea is simply to run RT on the sample at the nth stage and output the result as the nth guess.

Definition An operator RT_∞ from infinite sequences of skeletons $s_1, s_2, s_3, \ldots$ to infinite sequences of skeletal tree automata $A_1, A_2, A_3, \ldots$ is defined by

$$A_n = RT(\{s_1, s_2, \ldots, s_n\}) \qquad \text{for all } n \geq 1.$$

We need to show that this converges to a correct guess after a finite number of stages.

Definition An infinite sequence of skeletons $s_1, s_2, s_3, \ldots$ is defined to be a *positive presentation* of a skeletal tree automaton A if and only if the set $\{s_1, s_2, s_3, \ldots\}$ is precisely $T(A)$. An infinite sequence of skeletal tree automata $A_1, A_2, A_3, \ldots$ is said to *converge to* a skeletal tree automaton A if and only if there exists an integer N such that for all $n \geq N$, A_n is isomorphic to A.

The following result is necessary for the proof of correct identification in the limit of the reversible skeletal tree automata from positive presentation. We extend δ to $(V \cup Q)^T$ by letting $\delta(q) = q$ for $q \in Q$, where Q is considered as a set of terminal symbols. In this definition, if $q = \delta(u)$ for $q \in Q$ and $u \in V^T$, then $\delta(t\#q) = \delta(t\#u)$ for $t \in V_\$^T$.

Proposition 16 *For any reversible skeletal tree automaton $A = (Q, Sk \cup \Sigma, \delta, \{q_f\})$, there effectively exists a characteristic sample.*

Proof. Clearly, if $T(A) = \emptyset$, then $CS = \emptyset$ is a characteristic sample for A. Suppose $T(A) \neq \emptyset$. For each state $q \in Q$, let $u(q)$ be a tree of the minimum size in $Sub(T(A))$ such that $\delta(u(q)) = q$, and $v(q)$ be a tree of the minimum size in $Sc(T(A))$ such that $\delta(v(q)\#q) = q_f$. For each $a \in \Sigma$, let $u(a) = a$. Let CS consist of all skeletons of the form $v(q)\#u(q)$ such that $q \in Q$ and all skeletons of the form $v(q)\#\sigma(u(q_1), \ldots, u(q_k))$ such that $q_1, \ldots, q_k \in Q \cup \Sigma$, $\sigma \in Sk_k$ and $q = \delta_k(\sigma, q_1, \ldots, q_k)$. It is clear that $CS \subseteq T(A)$. We show that CS is a characteristic sample for A.

Let A' be any reversible skeletal tree automaton such that $T(A') \supseteq CS$. We show that $U_{T(A')}(t) = U_{T(A')}(u(q))$ for all skeletons $t \in Sub(T(A))$, where $q = \delta(t)$. We prove it by induction on the depth of t. Suppose first that the depth of t is 0, i.e. $t = a \in \Sigma$.

Since $u(a) = a$, it holds for the depth 0. Next suppose that this holds for all skeletons of depth at most h, for some $h \geq 0$. Let t be a skeleton of depth $h + 1$ from $Sub(T(A))$, so that $t = \sigma(s_1, \ldots, s_k)$ for some skeletons $s_1, \ldots, s_k \in Sub(T(A))$ with depth at most h. By the induction hypothesis, $U_{T(A')}(s_i) = U_{T(A')}(u(q_i))$, where $q_i = \delta(s_i)$ for $1 \leq i \leq k$. Thus, $U_{T(A')}(t) = U_{T(A')}(\sigma(s_1, \ldots, s_k)) = U_{T(A')}(\sigma(u(q_1), s_2, \ldots, s_k)) = \cdots = U_{T(A')}(\sigma(u(q_1), \ldots, u(q_{k-1}), s_k)) = U_{T(A')}(\sigma(u(q_1), \ldots, u(q_k)))$. If $q' = \delta_k(\sigma, q_1, \ldots, q_k) = \delta(t)$, then $v(q')\#u(q')$ and $v(q')\#\sigma(u(q_1), \ldots, u(q_k))$ are both elements of CS. So $v(q')\#u(q')$, $v(q')\#\sigma(u(q_1), \ldots, u(q_k)) \in T(A')$. By Lemma 4, $U_{T(A')}(\sigma(u(q_1), \ldots, u(q_k))) = U_{T(A')}(u(q'))$. Hence $U_{T(A')}(t) = U_{T(A')}(u(q'))$, which completes the induction.

Thus for every $t \in T(A)$, $U_{T(A')}(t) = U_{T(A')}(u(q_f))$. Since $v(q_f) = \$$, $u(q_f) \in CS$ and so $u(q_f) \in T(A')$. This implies that $\$ \in U_{T(A')}(u(q_f)) = U_{T(A')}(t)$. Thus $t = \$\#t \in T(A')$. Hence $T(A)$ is contained in $T(A')$. Therefore CS is a characteristic sample for A. Q.E.D.

Then we conclude the following result.

Theorem 17 *Let A be a reversible skeletal tree automaton, $s_1, s_2, s_3, \ldots$ be a positive presentation of A, and $A_1, A_2, A_3, \ldots$ be the output of RT_∞ on this input. Then $A_1, A_2, A_3, \ldots$ converges to the canonical skeletal tree automaton A' for $T(A)$.*

Proof. By Theorem 16, there exists a characteristic sample for A. Let N be sufficiently large that the set $\{s_1, s_2, \ldots, s_N\}$ contains a characteristic sample for A. For any reversible skeletal tree automaton A', $T(A') \supseteq \{s_1, s_2, \ldots, s_n\}$ implies $T(A_n) \subseteq T(A')$, by the definition of RT_∞ and Theorem 14. Thus for $n \geq N$, $T(A_n) = T(A)$, by the definition of a characteristic sample. Moreover it is easily checked that the skeletal tree automaton output by RT is stripped, and therefore canonical, by Lemma 11. Hence A_n is isomorphic to $C(T(A))$ for all $n \geq N$, so $A_1, A_2, A_3, \ldots$ converges to $C(T(A))$. Q.E.D.

We may modify RT by a simple updating scheme to have good incremental behavior so that A_{n+1} may be obtained from A_n and s_{n+1}.

Input : a nonempty positive structural sample Sa;
Output : a reversible context-free grammar G;
Procedure :
Run RT on the sample Sa;
Let $G = G'(RT(Sa))$ and output the grammar G.

Figure 3: The learning algorithm RC for Reversible Grammars

6.5 The Learning Algorithm RC for Context-Free Grammars

In this section, we describe and analyze the algorithm RC using the algorithm RT to learn reversible context-free grammars from positive samples of structural descriptions.

A *positive structural sample* of a context-free grammar G is a finite subset of $K(D(G))$. A positive structural sample CS of a reversible context-free grammar G is a *characteristic structural sample* for G if and only if for any reversible context-free grammar G', $K(D(G')) \supseteq CS$ implies $K(D(G)) \subseteq K(D(G'))$.

The input to RC is a finite nonempty set of skeletons Sa. The output is a particular reversible context-free grammar $G = RC(Sa)$ whose characteristic structural sample is precisely Sa. The learning algorithm RC is illustrated in Figure 3.

The following propositions and theorems of the correctness, time complexity and correct structural identification in the limit of the algorithm RC are straightforwardly derived by using Proposition 5 from the corresponding results for the algorithm RT described in Sections 6.2, 6.3 and 6.4.

Theorem 18 *Let Sa be a nonempty positive structural sample of skeletons, and G_f be the output of the context-free grammar by the algorithm RC on input Sa. Then G_f is reversible and for any reversible context-free grammar G, $K(D(G)) \supseteq Sa$ implies $K(D(G_f)) \subseteq K(D(G))$.*

Theorem 19 *The algorithm RC may be implemented to run in time polynomial in the sum of the sizes of the input skeletons.*

Define an operator RC_∞ from infinite sequences of skeletons $s_1, s_2, s_3, \ldots$ to infinite sequences of context-free grammars $G_1, G_2, G_3, \ldots$ by

$$G_n = RC(\{s_1, s_2, \ldots, s_n\}) \qquad \text{for all } n \geq 1.$$

An infinite sequence of skeltons $s_1, s_2, s_3, \ldots$ is defined to be a *positive structural presentation* of a context-free grammar G if and only if the set $\{s_1, s_2, s_3, \ldots\}$ is precisely $K(D(G))$. An infinite sequence of context-free grammars $G_1, G_2, G_3, \ldots$ is said to *converge to* a context-free grammar G if and only if there exists an integer N such that for all $n \geq N$, G_n is isomorphic to G.

Proposition 20 *For any reversible context-free grammar G, there effectively exists a characteristic structural sample.*

Now we have the following.

Theorem 21 *Let G be a reversible context-free grammar, $s_1, s_2, s_3, \ldots$ be a positive structural presentation of G, and $G_1, G_2, G_3, \ldots$ be the output of RC_∞ on this input. Then $G_1, G_2, G_3, \ldots$ converges to a reversible context-free grammar G' such that $K(D(G')) = K(D(G))$.*

We modify the algorithm RC to learn extended reversible context-free grammars from positive samples of their structural descriptions.

We can easily verify that given a positive structural presentation of an extended reversible context-free grammar G, the algorithm RC', illustrated in Figure 4, converges to an extended reversible context-free grammar which is structurally equivalent to G and runs in time polynomial in the sum of the sizes of the input skeltons. This implies that if information on the structure of the grammar in the form of extended reversible is available to the learning algorithm, the full class of context-free languages can be learned efficiently from positive samples.

Input : a nonempty positive structural sample Sa;
Output : an extended reversible context-free grammar G;
Procedure :
Let $Sa' = Sa - \{\sigma(a) \mid a \in \Sigma\}$;
Let $Uni = Sa \cap \{\sigma(a) \mid a \in \Sigma\}$;
Run RC on the sample Sa' and let $G' = (N, \Sigma, P, S)$ be $RC(Sa')$;
Let $P' = \{S \rightarrow a \mid \sigma(a) \in Uni\}$;
Let $G = (N, \Sigma, P \cup P', S)$ and output the grammar G.

Figure 4: The learning algorithm RC' for Extended Reversible Grammars

7 Example Runs

In the process of learning context-free grammars from their structural descriptions, the problem is to reconstruct the nonterminal labels because the set of derivation trees of the unknown context-free grammar is given with all nonterminal labels erased.

The structural descriptions of a context-free grammar can be equivalently represented by means of the parenthesis grammar. For example, the structural description in Figure 1 can be represented as the following sentence of the parenthesis grammar:

$$\langle\ \langle \text{ the } \langle \text{ big dog } \rangle\ \rangle\ \langle \text{ chases } \langle \text{ a } \langle \text{ young girl } \rangle\ \rangle\ \rangle\ \rangle$$

In the following, we demonstrate three examples to show the learning process of the algorithm RC. Three kinds of grammars will be learned, the first is a context-free grammar for a simple natural language, the second is a context-free grammar for a subset of the syntax for a programming language Pascal, and the third is an inherently ambiguous context-free grammar.

7.1 Simple Natural Language

Now suppose that the learning algorithm RC is going to learn the following unknown context-free grammar G_U for a simple natural language:

$Sentence \rightarrow Noun\_phrase\ Verb\_phrase$
$Noun\_phrase \rightarrow Determiner\ Noun\_phrase2$
$Noun\_phrase2 \rightarrow Noun$
$Noun\_phrase2 \rightarrow Adjective\ Noun\_phrase2$
$Verb\_phrase \rightarrow Verb\ Noun\_phrase$
$Determiner \rightarrow$ the
$Determiner \rightarrow$ a
$Noun \rightarrow$ girl
$Noun \rightarrow$ cat
$Noun \rightarrow$ dog
$Adjective \rightarrow$ young
$Verb \rightarrow$ likes
$Verb \rightarrow$ chases.

First suppose that the learning algorithm RC is given the sample:

$$\langle\ \langle\ \langle \text{the} \rangle\ \langle\ \langle \text{girl} \rangle\ \rangle\ \rangle\ \langle\ \langle \text{likes} \rangle\ \langle\ \langle \text{a} \rangle\ \langle\ \langle \text{cat} \rangle\ \rangle\ \rangle\ \rangle\ \rangle$$
$$\langle\ \langle\ \langle \text{the} \rangle\ \langle\ \langle \text{girl} \rangle\ \rangle\ \rangle\ \langle\ \langle \text{likes} \rangle\ \langle\ \langle \text{a} \rangle\ \langle\ \langle \text{dog} \rangle\ \rangle\ \rangle\ \rangle\ \rangle$$

RC first constructs the base context-free grammar for them. However it is not reversible. So RC merges distinct nonterminals repeatedly and outputs the following reversible context-free grammar:

$S \rightarrow NT1\ NT2$
$NT1 \rightarrow NT3\ NT4$
$NT4 \rightarrow NT5$
$NT2 \rightarrow NT6\ NT7$
$NT7 \rightarrow NT8\ NT9$
$NT9 \rightarrow NT10$
$NT3 \rightarrow$ the
$NT5 \rightarrow$ girl
$NT6 \rightarrow$ likes
$NT8 \rightarrow$ a
$NT10 \rightarrow$ cat
$NT10 \rightarrow$ dog.

RC has learned that "cat" and "dog" belong to the same syntactic category. However RC has not learned that "girl" belongs to the same syntactic category (*noun*) as "cat" and "dog", and "a" and "the" belong to the same syntactic category (*determiner*). Suppose that in the next stage the following examples are added to the sample:

$$\langle\ \langle\ \langle \text{a} \rangle\ \langle\ \langle \text{dog} \rangle\ \rangle\ \rangle\ \langle\ \langle \text{chases} \rangle\ \langle\ \langle \text{the} \rangle\ \langle\ \langle \text{girl} \rangle\ \rangle\ \rangle\ \rangle\ \rangle$$
$$\langle\ \langle\ \langle \text{a} \rangle\ \langle\ \langle \text{dog} \rangle\ \rangle\ \rangle\ \langle\ \langle \text{chases} \rangle\ \langle\ \langle \text{a} \rangle\ \langle\ \langle \text{cat} \rangle\ \rangle\ \rangle\ \rangle\ \rangle$$

Then RC outputs the reversible context-free grammar:

$$\begin{array}{l} S \rightarrow NT1\ NT2 \\ NT1 \rightarrow NT3\ NT4 \\ NT4 \rightarrow NT5 \\ NT2 \rightarrow NT6\ NT1 \\ NT1 \rightarrow NT7\ NT8 \\ NT8 \rightarrow NT9 \\ NT3 \rightarrow \text{the} \\ NT5 \rightarrow \text{girl} \\ NT6 \rightarrow \text{likes} \\ NT6 \rightarrow \text{chases} \\ NT7 \rightarrow \text{a} \\ NT9 \rightarrow \text{cat} \\ NT9 \rightarrow \text{dog.} \end{array}$$

RC has learned that "likes" and "chases" belong to the same syntactic category (*verb*) and "the girl", "a dog" and "a cat" are identified as the same phrase (*noun_phrase*). However RC has not learned yet that "a" and "the" belong to the same syntactic category. Suppose that in the further stage the following examples are added to the sample:

$$\langle\ \langle\ \langle\text{a}\rangle\ \langle\ \langle\text{dog}\rangle\ \rangle\ \rangle\ \langle\ \langle\text{chases}\rangle\ \langle\ \langle\text{a}\rangle\ \langle\ \langle\text{girl}\rangle\ \rangle\ \rangle\ \rangle\ \rangle$$

$$\langle\ \langle\ \langle\text{the}\rangle\ \langle\ \langle\text{dog}\rangle\ \rangle\ \rangle\ \langle\ \langle\text{chases}\rangle\ \langle\ \langle\text{a}\rangle\ \langle\ \langle\text{young}\rangle\ \langle\ \langle\text{girl}\rangle\ \rangle\ \rangle\ \rangle\ \rangle\ \rangle$$

RC outputs the reversible context-free grammar:

$$\begin{array}{l} S \rightarrow NT1\ NT2 \\ NT1 \rightarrow NT3\ NT4 \\ NT4 \rightarrow NT5 \\ NT4 \rightarrow NT6\ NT4 \\ NT2 \rightarrow NT7\ NT1 \\ NT3 \rightarrow \text{the} \\ NT3 \rightarrow \text{a} \\ NT5 \rightarrow \text{girl} \\ NT5 \rightarrow \text{cat} \\ NT5 \rightarrow \text{dog} \\ NT6 \rightarrow \text{young} \\ NT7 \rightarrow \text{likes} \\ NT7 \rightarrow \text{chases.} \end{array}$$

This grammar is isomorphic to the unknown grammar G_U.

7.2 Programming Language

Suppose that the learning algorithm RC is going to learn the following unknown context-free grammar G_U for a subset of the syntax for a programming language Pascal:

$$\begin{array}{l}
Statement \rightarrow v \ := \ Expression \\
Statement \rightarrow \text{while } Condition \text{ do } Statement \\
Statement \rightarrow \text{if } Condition \text{ then } Statement \\
Statement \rightarrow \text{begin } Statementlist \text{ end} \\
Statementlist \rightarrow Statement \ ; \ Statementlist \\
Statementlist \rightarrow Statement \\
Condition \rightarrow Expression \ > \ Expression \\
Expression \rightarrow Term \ + \ Expression \\
Expression \rightarrow Term \\
Term \rightarrow Factor \\
Term \rightarrow Factor \ \times \ Term \\
Factor \rightarrow v \\
Factor \rightarrow (\ Expression \ ).
\end{array}$$

First suppose that RC is given the sample:

$$\langle \ v \ := \ \langle \ \langle\langle v\rangle\rangle \ + \ \langle\langle\langle v\rangle\rangle\rangle \ \rangle \ \rangle$$

$$\langle \ v \ := \ \langle\langle \ \langle v\rangle \ \times \ \langle\langle v\rangle\rangle \ \rangle\rangle \ \rangle$$

$$\langle \ v \ := \ \langle \ \langle\langle v\rangle\rangle \ + \ \langle\langle \ \langle v\rangle \ \times \ \langle\langle v\rangle\rangle \ \rangle\rangle \ \rangle \ \rangle$$

$$\langle \ v \ := \ \langle\langle \ \langle \ '(' \ \langle \ \langle\langle v\rangle\rangle \ + \ \langle\langle\langle v\rangle\rangle\rangle \ \rangle \ ')' \ \rangle \ \times \ \langle\langle v\rangle\rangle \ \rangle\rangle \ \rangle$$

RC outputs the following reversible context-free grammar which generates the set of all assignment statements whose right-hand sides are arithmetic expressions consisting of a variable "v", the operations of addition "+" and multiplication "×" and the pair of parentheses '(' and ')':

$$\begin{array}{l}
S \rightarrow v \ := NT1 \\
NT1 \rightarrow NT2 \\
NT1 \rightarrow NT2 \ + \ NT1 \\
NT2 \rightarrow NT3 \\
NT2 \rightarrow NT3 \ \times \ NT2 \\
NT3 \rightarrow v \\
NT3 \rightarrow (\ NT1 \ ).
\end{array}$$

Next suppose that RC is given four more examples:

$$\langle \text{ while } \langle \ \langle\langle\langle v\rangle\rangle\rangle \ > \ \langle\langle\langle v\rangle \ \times \ \langle\langle v\rangle\rangle\rangle\rangle \ \rangle \text{ do } \langle \ v \ := \ \langle\langle\langle v\rangle\rangle \ + \ \langle\langle\langle v\rangle\rangle\rangle\rangle \ \rangle \ \rangle$$

$$\langle \text{ if } \langle \ \langle\langle\langle v\rangle\rangle\rangle \ > \ \langle\langle\langle v\rangle \ \times \ \langle\langle v\rangle\rangle\rangle\rangle \ \rangle \text{ then } \langle \ v \ := \ \langle\langle\langle v\rangle\rangle \ + \ \langle\langle\langle v\rangle\rangle\rangle\rangle \ \rangle \ \rangle$$

$$\langle \text{ begin } \langle \ \langle v \ := \ \langle\langle\langle v\rangle\rangle \ + \ \langle\langle\langle v\rangle\rangle\rangle\rangle\rangle \ ; \ \langle \ \langle v \ := \ \langle\langle\langle v\rangle \ \times \ \langle\langle v\rangle\rangle\rangle\rangle\rangle \ \rangle \ \rangle \text{ end } \rangle$$

$$\langle \text{ begin } \langle \ \langle v \ := \ \langle\langle\langle v\rangle \ \times \ \langle\langle v\rangle\rangle\rangle\rangle\rangle \ \rangle \text{ end } \rangle$$

RC outputs the following reversible context-free grammar isomorphic to the unknown grammar G_U:

$$
\begin{array}{l}
S \rightarrow v \ := NT1 \\
S \rightarrow \text{while } NT4 \text{ do } S \\
S \rightarrow \text{if } NT4 \text{ then } S \\
S \rightarrow \text{begin } NT5 \text{ end} \\
NT1 \rightarrow NT2 \\
NT1 \rightarrow NT2 \ + \ NT1 \\
NT2 \rightarrow NT3 \\
NT2 \rightarrow NT3 \ \times \ NT2 \\
NT3 \rightarrow v \\
NT3 \rightarrow (\ NT1\ ) \\
NT4 \rightarrow NT1 \ > \ NT1 \\
NT5 \rightarrow S \\
NT5 \rightarrow S\ ;\ NT5.
\end{array}
$$

7.3 Inherently Ambiguous Language

Suppose that the learning algorithm RC is going to learn the following unknown context-free grammar G_U for the language $\{a^m b^m c^n d^n \mid m \geq 1, n \geq 1\} \cup \{a^m b^n c^n d^m \mid m \geq 1, n \geq 1\}$ which is known as an inherently ambiguous context-free language:

$$
\begin{array}{l}
S \rightarrow A\ B \\
S \rightarrow a\ C\ d \\
A \rightarrow a\ b \\
A \rightarrow a\ A\ b \\
B \rightarrow c\ d \\
B \rightarrow c\ B\ d \\
C \rightarrow D \\
D \rightarrow a\ D\ d \\
D \rightarrow E \\
E \rightarrow b\ c \\
E \rightarrow b\ E\ c.
\end{array}
$$

First suppose that RC is given the sample:

$$\langle\ \langle a\ b\rangle\ \langle c\ d\rangle\ \rangle$$

$$\langle\ \langle a\ \langle a\ b\rangle\ b\rangle\ \langle c\ \langle c\ d\rangle\ d\rangle\ \rangle$$

$$\langle\ \langle a\ b\rangle\ \langle c\ \langle c\ d\rangle\ d\rangle\ \rangle$$

RC outputs the following reversible context-free grammar which generates the language $\{a^m b^m c^n d^n \mid m \geq 1, n \geq 1\}$:

$$
\begin{array}{l}
S \to NT1\ NT2 \\
NT1 \to a\ b \\
NT1 \to a\ NT1\ b \\
NT2 \to c\ d \\
NT2 \to c\ NT2\ d.
\end{array}
$$

Next suppose that RC is given three more examples:

$$\langle a\ \langle\ \langle\ \langle b\ c\rangle\ \rangle\ \rangle\ d\rangle$$

$$\langle a\ \langle\ \langle a\ \langle\ \langle b\ \langle b\ c\rangle\ c\rangle\ \rangle\ d\rangle\ \rangle\ d\rangle$$

$$\langle a\ \langle\ \langle\ \langle b\ \langle b\ c\rangle\ c\rangle\ \rangle\ \rangle\ d\rangle$$

RC outputs the following reversible context-free grammar isomorphic to the unknown grammar G_U:

$$
\begin{array}{l}
S \to NT1\ NT2 \\
S \to a\ NT3\ d \\
NT1 \to a\ b \\
NT1 \to a\ NT1\ b \\
NT2 \to c\ d \\
NT2 \to c\ NT2\ d \\
NT3 \to NT4 \\
NT4 \to NT5 \\
NT4 \to a\ NT4\ d \\
NT5 \to b\ c \\
NT5 \to b\ NT5\ c.
\end{array}
$$

8 Concluding Remarks

In this paper, we have considered the problem of learning context-free grammars from positive samples of their structural descriptions and investigated the effect of assuming example presentations in the form of structural descriptions on learning from positive samples. By introducing the class of reversible context-free grammars, we have shown that the assumption of examples in the form of structural descriptions makes it possible to learn the full class of context-free languages from positive samples and in polynomial time. Thus this problem setting makes our learning algorithm practical and useful.

Angluin [2] has taken an entirely different approach with the same motivation of investigating what assumption can compensate for the lack of explicit negative information in positive samples and studied the effect of assuming randomly drawn examples on various types of limiting identification of formal languages. She showed that in her criterion for limit identification analogous to Valiant's finite criterion [17], the assumption of stochastically generated examples does not enlarge the class of learnable sets of formal languages from positive samples. Compared this result with ours in this paper, we can conclude that the assumption of examples in the form of structural descriptions strongly compensates for the lack of explicit negative information in positive samples and is helpful for efficient learning of context-free grammars.

Lastly we remark on related work. Crespi-Reghizzi [9] is most closely related, as it describes a constructive method for learning context-free grammars from positive samples of structural descriptions. However his algorithm and our one use completely different methods and learn different classes of context-free grammars. The class of reversible context-free grammars can generate all of the context-free languages, while his class of context-free grammars defines a subclass of context-free languages, called *noncounting context-free languages* [10]. Since our formalization is based on tree automata, one of merits of our method is the simplicity of the theoretical analysis and the easiness of understanding the algorithm, whereas the time efficiency of his algorithm [9] is still not clear.

Acknowledgements

The author is indebted to Dr. Kaname Kobayashi for his useful suggestion and warm encouragement. He is very grateful to Yuji Takada, his colleague, and Dr. Takashi Yokomori, University of Electro-Communications, who worked through an earlier draft of the paper and many comments. The author would also like to thank Shigemi Ooizumi for implementing this algorithm and exhibiting it works well and fast.

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] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1983.

[2] D. Angluin. *Identifying languages from stochastic examples.* RR 614, YALEU/DCS, 1988.

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

[4] D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29:741–765, 1982.

[5] D. Angluin. *Learning k-bounded context-free grammars.* RR 557, YALEU/DCS, 1987.

[6] D. Angluin. Learning regular sets from queries and counter-examples. *Information and Computation*, 75:87–106, 1987.

[7] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[8] P. Berman and R. Roos. Learning one-counter languages in polynomial time. In *Proceedings of IEEE FOCS '87*, pages 61–67, 1987.

[9] S. Crespi-Reghizzi. An effective model for grammar inference. In B. Gilchrist, editor, *Information Processing 71*, pages 524–529, Elsevier North-Holland, 1972.

[10] S. Crespi-Reghizzi, G. Guida, and D. Mandrioli. Noncounting context-free languages. *Journal of the ACM*, 25:571–580, 1978.

[11] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[12] J. N. Gray and M. A. Harrison. On the covering and reduction problems for context-free grammars. *Journal of the ACM*, 19:675–698, 1972.

[13] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. In *Proceedings of 1st Workshop on Computational Learning Theory*, pages 42–55, 1988.

[14] O. H. Ibarra and T. Jiang. Learning regular languages from counterexamples. In *Proceedings of 1st Workshop on Computational Learning Theory*, pages 371–385, 1988.

[15] L. S. Levy and A. K. Joshi. Skeletal structural descriptions. *Information and Control*, 39:192–211, 1978.

[16] Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. In *Proceedings of 1st Workshop on Computational Learning Theory*, pages 330–344, 1988. To appear in *Theoretical Computer Science*.

[17] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.