ICOT Technical Memorandum: TM-0506

TM-0506

An Efficient Learning of Context-Free Grammars for Bottom-Up Parsers

by

Y. Sakakibara (Fujitsu)

May, 1988

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 for Bottom-Up Parsers[1]

Yasubumi SAKAKIBARA

International Institute for Advanced Study of Social Information Science,
FUJITSU LIMITED, 140 Miyamoto, Numazu, Shizuoka 410-03, JAPAN

abstract

We consider the problem of learning a context-free grammar from examples. In this paper, the problem is slightly different from the usual grammatical inference problem. The problem is to learn a context-free grammar adequate for bottom-up parsing or designing bottom-up parser. Our final goal is to present the system using grammatical inference methods to develop a grammar for bottom-up parsing. Then what do we mean by "a grammar adequate for bottom-up parsing". Our answer is that the grammar should have the intended structure for parsing and allow the process of bottom-up parsing to be made easily. Furthermore for a practical use, we require that the grammar should be learned from positive-only examples and the grammar should be learned efficiently. To achieve those requirements, we present an efficient algorithm for learning a context-free grammar from positive examples of structural descriptions. Structural descriptions of a context-free grammar are unlabelled parse trees of the grammar, the *shapes* of parse trees. Thus the input to the learning algorithm is a finite set of shapes of parse trees. We show that the learning algorithm learns a grammar which is structurally equivalent to the unknown grammar and achieves the polynomial time bound.

[1] This is part of the work in the major R&D of FGCP, conducted under program set up by MITI.

1 Introduction

We consider the problem of learning a context-free grammar from 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, a "correct" grammar only means a grammar which correctly generates the language. In this paper, the problem is defined to learn a context-free grammar adequate for bottom-up parsing. Our final goal is to present the system using grammatical inference methods to develop (or design) a grammar for bottom-up parsing. In this problem setting, it is quite natural for us to require the grammar to be learned to have the following properties.

- *The learned grammar should have the intended structure.* The traditional grammatical inference problem is defined to identify a grammar G from examples of the unknown language L such that G correctly generates the language L, i.e., $L = L(G)$. However for any context-free language L there exist infinitely many grammars G such that $L = L(G)$. Furthermore, those grammars may have different structures. Consider the following example. The grammar G_1 below describes the set of all valid arithmetic expressions involving a variable "v" and the operations of multiplication "$\times$" and addition "$+$".

$$S \rightarrow v \mid Av$$

$$A \rightarrow v+ \mid v\times \mid v + A \mid v \times A$$

(the grammar G_1)

However the structure assigned by the grammar G_1 to sentences is semantically meaningless. The same language can be specified by the grammar G_2 below in a meaningful manner.

$$S \rightarrow E$$

$$E \rightarrow F \mid F + E$$

$$F \rightarrow v \mid v \times F$$

(the grammar G_2)

Here the phrases are all significant in terms of the rules of arithmetic. Although G_1 and G_2 are equivalent (i.e. $L(G_1) = L(G_2)$), 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 assigns the structures to the sentences in a nonsignificant manner. Thus in the problem of learning a grammar for parsing, the structure of the learned grammar is more significant because the learned grammar is intended for use in a practical situation entailing the translation or interpretation of sentences in a compiler. However in the framework of the usual grammatical inference, it is impossible to infer such a grammar (e.g. not the grammar G_1 but G_2) which has the correct (intended) structure. To do so, it is necessary for us to assume that information on the structure of the grammar is available to the learning algorithm. This hypothesis is in agreement with studies on natural language by Chomsky in terms of the theory of phrase structure grammars which claim that the availability of structural descriptions is prerequisite for language description, since there must be a partially semantic basis in syntax acquisition. In the case of context-free grammars, the structure of grammars is usually described by the *shapes* of the parse trees, called *structural descriptions.* A structural description is a kind of tree whose interior nodes have no label. The algorithm that we present learns a context-free grammar which has the intended structure from structural descriptions.

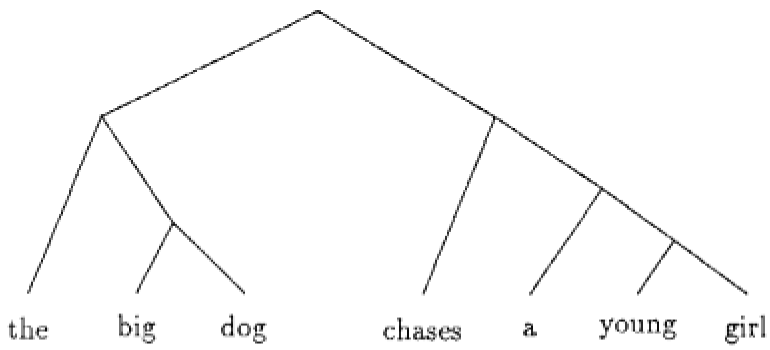

Figure 1: A structural description for "the big dog chases a young girl"

- *The learned grammar should allow the process of bottom-up parsing to be made easily.* Bottom-up parsing consists of (i) successively finding phrases and (ii) 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. The family of *invertible grammars*, which is one of normal forms for context-free grammars, allows reduction decisions to be made simply. A context-free grammar $G = (N, \Sigma, P, S)$ is said to be *invertible* if $A \to \alpha$ and $B \to \alpha$ in P implies $A = B$. Thus invertible grammars have unique righthand sides of the productions and the reduction phase of parsing becomes a matter of table lookup. The motivation for studying invertible grammars comes from the theory of bottom-up parsing. Since the invertible grammar is a normal form, for any context-free language L, there exists an invertible grammar which generates L. Further for any context-free grammar G, there exists an invertible grammar which has the same structure as G and generates the same language as G (i.e. which is *structurally equivalent* to G). The algorithm that we present learns an invertible context-free grammar from structural descriptions.

- *The grammar should be learned from positive-only examples.* In the case of learning an unknown language L, there is a fundamental, important distinction between giving only positive information (members of L) and giving both positive and negative information (both members and nonmembers of L). A *positive presentation* of L is an infinite sequence giving all and only the elements of L. A *complete presentation* of L is a sequence of ordered pairs $\langle w, d\rangle$ from $\Sigma^* \times \{0, 1\}$ such that $d = 1$ iff 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. Intuitively, an added difficulty in trying to learn from positive rather than complete presentation is the problem of "overgeneralization". Learning from positive presentation is strictly less powerful than learning from complete presentation. Gold [7] shows that any set of languages containing all the finite languages and at least one infinite language cannot be identified in the limit from positive presentations. This result applies to many important classes of languages (e.g., the regular languages and the context-free languages). However Angluin [1] gives a characterization of the sets of recursive languages that can be identified in the limit from positive presentation. In this paper, we consider the problem of learning a context-free grammar from positive presentation because assuming the teacher giving positive information of the grammar is acceptable in a practical use, whereas assuming the teacher giving complete information of it is not so easy for users. Since, in our problem setting, information of the grammar is the structural descriptions of it, it is assumed that positive presentation of structural descriptions is given to the learning algorithm. As we said before, the class of context-free grammars cannot be identified from positive presentation. We define a subclass of context-free grammars, called *reversible context-free grammars*, that is still powerful to define usual languages, and show that the class of reversible context-free grammars can be identified from positive presentation of structural descriptions.

- *The grammar should be learned efficiently.* In practical use of the grammatical inference, the crucial point is the time efficiency of the learning algorithm. One of criteria for evaluating the time efficiency of the learning algorithm is the polynomial time bound. Several learning algorithms for different domains [3,8] have been studied to achieve the polynomial time bound. We investigate an algorithm for learning a context-free grammar from structural descriptions in polynomial time. It is known that the set of parse trees of a context-free grammar constitutes a rational set of trees, where a *rational set* of trees is a set of trees which can be recognized by some tree automaton. Further the set of structural descriptions of a context-free grammar also constitutes a rational set of trees. Based on this observation, the problem of learning a context-free grammar from structural descriptions can be reduced to the problem of learning a tree automaton. Then by extending various existing efficient learning algorithms for finite automata to the ones for tree automata, we can get various efficient learning algorithms for context-free grammars. In this paper, we extend Angluin's efficient algorithm [2] for learning a finite automaton from positive presentation and present an efficient algorithm for learning a context-free grammar from positive presentation of structural descriptions.

2 Preliminaries

A *ranked alphabet* V is a finite set of symbols associated with a finite relation called the *rank* relation $r_V \subseteq V \times \{0,1,2,\ldots,m\}$. V_n denotes the subset $\{f \in V \mid (f,n) \in r_V\}$ of V. Especially, we call V_0, denoted Σ (i.e. $\Sigma = V_0$), the *terminal alphabet.* In many cases the symbols in V_n are 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.* A *tree* over V is a mapping $t : Dom_t \mapsto V$, which labels the nodes of the tree domain Dom_t. V^T denotes the set of all trees over V. A *tree language* is any subset of V^T. A *terminal node* in Dom_t is one which has no descendant. For a set of trees T, the set of subtrees of elements of T is denoted by $Sub(T)$.

A *(deterministic frontier-to-root) tree automaton* 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:

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

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

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 :

$$\delta(f(t_1,\ldots,t_k)) = \begin{cases} \delta_k(f, \delta(t_1), \ldots, \delta(t_k)) & \text{if } k > 0, \\ \delta_0(f) & \text{if } k = 0. \end{cases}$$

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

If δ is a state transition function from $V_k \times (Q \cup V_0)^k$ to 2^Q $(k = 1, 2, \ldots, m)$, then the tree automaton is *nondeterministic.* For a nondeterministic tree automaton $NA = (Q, V, \delta, F)$, we define $T(NA)$ as follows. δ can be extended to V^T by letting :

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

Then the set $T(NA)$ of trees accepted by NA is defined as $T(NA) = \{t \in V^T \mid \delta(t) \cap F \neq \phi\}$. Note that nondeterministic tree automata are no more powerful than deterministic tree automata.

Let A be a tree automaton which accepts a set of trees T. A is *minimum* iff A has the minimum number of states among all tree automata which accept T. The minimum tree automaton is unique up to isomorphism [4].

A *context-free grammar* is denoted $G = (N, \Sigma, P, S)$, where N and Σ are alphabets of *nonterminals* and *terminals* respectively such that $N \cap \Sigma = \phi$. 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.* If $A \rightarrow \beta$ is a production of P and α and γ are any strings in $(N \cup \Sigma)^*$, then $\alpha A \gamma \Rightarrow \alpha\beta\gamma$. $\overset{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$. The *language generated* by G, denoted $L(G)$, is $\{w \mid w \text{ is in } \Sigma^* \text{ and } S \overset{*}{\Rightarrow} w\}$. Two context-free grammars G_1 and G_2 are said to be *equivalent* if $L(G_1) = L(G_2)$. A *parenthesis grammar* is a context-free grammar $G = (N, \Sigma, P, S)$ such that the productions in P are restricted to the form $A \rightarrow \langle\alpha\rangle$, where $\langle$ and $\rangle$ are special symbols not in Σ and α contains neither $\langle$ nor $\rangle$. Without loss of generality, we restrict our consideration to only ϵ-free context-free grammars.

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 :

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

A tree in $D_A(G)$ is called a *parse tree* of G from A. For the set $D_S(G)$ of parse 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 $r_{Sk} \subseteq \{\sigma\} \times \{1, 2, 3, \ldots, m\}$. A tree defined over $Sk \cup \Sigma$ is called a *skeleton*. 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

$$s(t)(x) = \begin{cases} t(x) & \text{if } x \text{ is a terminal node,} \\ \sigma & \text{otherwise.} \end{cases}$$

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 which has a special symbol 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.

The *structural description* of a context-free grammar G is the skeletal set $K(D(G))$. Two context-free grammars G_1 and G_2 are said to be *structurally equivalent* if $K(D(G_1)) = K(D(G_2))$. Note that if G_1 and G_2 are structurally equivalent, they are equivalent, too.

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

Proposition 1 *Let $G = (N, \Sigma, P, S)$ be a context-free grammar. The corresponding non-deterministic tree automaton $NA(G) = (Q, Sk \cup \Sigma, \delta, F)$ 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 $a \in \Sigma$.

Then $T(NA(G)) = K(D(G))$. That is, the set of trees accepted by $NA(G)$ is equal to the structural description of G.

Proposition 2 *Let $A = (Q, Sk \cup \Sigma, \delta, F)$ be a tree automaton. The corresponding context-free grammar $G(A) = (N, \Sigma, P, S)$ is defined as follows.*

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

Then $K(D(G(A))) = T(A)$. That is, the structural description of $G(A)$ is equal to the set of trees accepted by A.

Hence the problem of learning a context-free grammar from structural descriptions can be reduced to the problem of learning a tree automaton. All following results for context-free grammars are derived by using Proposition 1 and 2 from the similar results for tree automata. Thus behind the theory for context-free grammars concluded in this paper, there always exists the corresponding theory for tree automata.

3 Structural Identification

Gold's theoretical study of language learning introduces a fundamental concept that is very important in inductive inference : *identification in the limit.* In the Gold's traditional definition, for an inductive inference algorithm IA that is attempting to learn the unknown language L, an infinite sequence of examples of L is presented. Then after some finite number of example presentations, IA guesses the correct conjecture of the language and never changes (converges to) its guess after this. In the case that the conjectures are in the form of grammars, IA identifies in the limit a grammar G such that $L(G) = L$.

On the other hand, as in [8], 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. Suppose G is the unknown 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. In this case, a sequence of examples of the language $L(G)$ is replaced by a sequence of examples of the structural description $K(D(G))$. 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). This type of identification is called *structural identification in the limit.*

By Proposition 1 and 2, the problem of structural identification of context-free grammars is reduced to the problem of identification of tree automata, and hence to the problem of identification of tree languages.

4 Condition for Positive Inference

In order to do correct identification in the limit from positive presentation, we must avoid the problem of "overgeneralization". Angluin has shown in [1] various conditions for inference from positive presentation that avoids overgeneralization. In her framework, the domain is a family of languages $\mathcal{L} = \{L_1, L_2, L_3 \ldots,\}$. A *positive sample* of the language L is a finite subset of L. One of conditions for inference from positive presentation is following.

Condition-1 A family of language *satisfies Condition*-1 iff there exists an effective procedure which on any input $i \geq 1$ enumerates a positive sample S_i of L_i such that (*) for all $j \geq 1$, if $S_i \subseteq L_j$ then L_j is not a proper subset of L_i.

This condition requires that for every language L_i of the family $\mathcal{L}$, there exists a "telltale" finite subset S_i of L_i such that no language of the family $\mathcal{L}$ that also contains S_i is a proper subset of L_i.

These discussions and formulations can be applied to the case of *inference of tree languages*, and hence to the structural identification.

5 Reversible Context-Free Grammars

A context-free grammar $G = (N, \Sigma, P, S)$ is said to be *invertible* iff $A \rightarrow \alpha$ and $B \rightarrow \alpha$ in P implies $A = B$. 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 $L(G) = L$. A context-free grammar $G = (N, \Sigma, P, S)$ is *reset-free* iff 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 $B = C$. A context-free grammar G is said to be *reversible* iff G is invertible and reset-free. A context-free language L is defined to be *reversible* iff there exists a reversible context-free grammar G such that $L = L(G)$.

The idea of the reversible context-free grammars comes from the "reversible automata" and "reversible languages" in [1]. Basically, the corresponding tree automata for reversible context-free grammars are the extensions of "zero-reversible automata".

We now consider characteristic structural samples for the reversible context-free grammars. 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 iff for any reversible context-free grammar G', $K(D(G')) \supseteq CS$ implies $K(D(G)) \subseteq K(D(G'))$. The following result is necessary for the proof of correct identification in the limit of the reversible context-free grammars from positive presentation of structural descriptions.

Proposition 3 *For any reversible context-free grammar G, there exists a characteristic structural sample.*

6 The Reversible Learning Algorithms

In this section we describe and analyze the algorithm RC to learn a reversible context-free grammar from positive structural samples.

The input to RC is a finite nonempty set of skeletons Sa. The output is a particular reversible context-free grammar $G = RC(Sa)$. The learning algorithm RC begins with the primitive context-free grammar for Sa and generalizes it by merging nonterminals.

A *partition* of some set X is a set of pairwise disjoint nonempty subsets of X whose union is X. If π is a partition of X, then for any element $x \in X$ there is a unique element of π containing x, which we denote $Bl(x, \pi)$ and call the *block* of π containing x. A partition π is $finer$ than another partition π' iff every block of π' is a union of blocks of π. The *trivial partition* of a set X is the class of all sets $\{x\}$ such that $x \in X$.

Let $G = (N, \Sigma, P, S)$ be any context-free grammar. If π is any partition of N, we define the context-free grammar $G/\pi = (N', \Sigma, P', S')$ induced by π as follows. N' is the set of blocks of π (i.e. $N' = \pi$). S' is the block of π that contains S. The production $Bl \rightarrow Bl_1 \cdots Bl_k$ is in P' whenever there exist $A \in Bl$ and $A_i \in Bl_i \in \pi$ or $A_i = Bl_i \in \Sigma$ for $1 \leq i \leq k$ such that $A \rightarrow A_1 \cdots A_k$ is in P.

Let Sa be a finite set of skeletons. Define the *primitive context-free grammar* for Sa,

$G(Sa) = (N, \Sigma, P, S)$, as follows :

$$
\begin{aligned}
N &= (Sub(Sa) - \Sigma) \cup \{S\}, \\
P &= \{\sigma(A_1, \ldots, A_k) \rightarrow A_1 \cdots A_k \mid \sigma(A_1, \ldots, A_k) \in N\} \\
&\quad \cup \{S \rightarrow A_1 \cdots A_k \mid \sigma(A_1, \ldots, A_k) \in Sa\}.
\end{aligned}
$$

Then $G(Sa)$ is a context-free grammar such that $K(D(G(Sa))) = Sa$.

Algorithm RC

Input : a nonempty positive structural sample Sa;

Output : a reversible context-free grammar G;

Procedure :

On input Sa, RC first constructs $G_0 = G(Sa)$, the primitive context-free grammar for Sa. It then constructs the finest partition π_f of the set N_0 of nonterminals of G_0 with the property that G_0/π_f is reversible, and outputs G_0/π_f.

To construct π_f, RC begins with the trivial partition of N_0 and repeatedly merges any two distinct blocks Bl_1 and Bl_2 if either of the following conditions is satisfied.

1. There exist two productions of the forms $A \rightarrow A_1 \cdots A_k$ and $A' \rightarrow A'_1 \cdots A'_k$ in P_0 such that $A \in Bl_1$ and $A' \in Bl_2$, and for $1 \leq j \leq k$, A_j and A'_j both are in the same block or are the same terminal symbols.
2. There exist two productions of the forms $A \rightarrow A_1 \cdots A_k$ and $A' \rightarrow A'_1 \cdots A'_k$ in P_0 and an integer l $(1 \leq l \leq k)$ such that $A_l \in Bl_1$ and $A'_l \in Bl_2$, A and A' are in the same block, and for $1 \leq j \leq k$, $j \neq l$, A_j and A'_j both are in the same block or are the same terminal symbols.

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

This completes the description of the algorithm RC, and we next analyze its correctness and time efficiency.

Theorem 4 *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 5 *The algorithm RC 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) is the number of symbols in its textual representation.*

Next we show that the algorithm RC may be used at the finite stages of an infinite learning process to identify the reversible context-free grammars in the limit from positive presentation of structural descriptions. The idea is simply to run RC on the sample at the nth stage and output the result as the nth guess. 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

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

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

An infinite sequence of skeletons $s_1, s_2, s_3, \ldots$ is defined to a *positive structural presentation* of a context-free grammar G iff 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 iff there exists an integer N such that for all $n \geq N$, G_n is isomorphic to G. By Proposition 3 and Theorem 4, we conclude the following result.

Theorem 6 *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 may modify RC by a simple updating scheme to have good incremental behavior so that G_{n+1} may be obtained from G_n and s_{n+1}.

7 An Example

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

The structural description 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 sentence of the parenthesis grammar:

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

Now suppose that the learning algorithm RC is going to learn the following unknown context-free grammar G_U for 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 primitive context-free grammar for them and assigns distinct nonterminal symbols to internal nodes. However it is not reversible. So RC merges distinct nonterminals repeatedly and produces the following reversible context-free grammar:

$$
\begin{aligned}
&S \to P\_1, P\_2.\\
&P\_1 \to P\_3, P\_4.\\
&P\_4 \to P\_5.\\
&P\_2 \to P\_6, P\_7.\\
&P\_7 \to P\_8, P\_9.\\
&P\_9 \to P\_10.\\
&P\_3 \to \text{the}.\\
&P\_5 \to \text{girl}.\\
&P\_6 \to \text{likes}.\\
&P\_8 \to \text{a}.\\
&P\_10 \to \text{cat}.\\
&P\_10 \to \text{dog}.
\end{aligned}
$$

RC learns that "cat" and "dog" belong to the same syntactic category. However *RC* does not learn that "girl" is the same syntactic category as "cat" and "dog" (*noun*), 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* produces the reversible context-free grammar:

$$
\begin{aligned}
&S \to P\_1, P\_2.\\
&P\_1 \to P\_3, P\_4.\\
&P\_4 \to P\_5.\\
&P\_2 \to P\_6, P\_1.\\
&P\_1 \to P\_7, P\_8.\\
&P\_8 \to P\_9.\\
&P\_3 \to \text{the}.\\
&P\_5 \to \text{girl}.\\
&P\_6 \to \text{likes}.\\
&P\_6 \to \text{chases}.\\
&P\_7 \to \text{a}.\\
&P\_9 \to \text{cat}.\\
&P\_9 \to \text{dog}.
\end{aligned}
$$

RC learns 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* does not still learn 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 produces the reversible context-free grammar:

$$
\begin{aligned}
&S \rightarrow P\_1, P\_2. \\
&P\_1 \rightarrow P\_3, P\_4. \\
&P\_4 \rightarrow P\_5. \\
&P\_4 \rightarrow P\_6, P\_4. \\
&P\_2 \rightarrow P\_7, P\_1. \\
&P\_3 \rightarrow \text{the}. \\
&P\_3 \rightarrow \text{a}. \\
&P\_5 \rightarrow \text{girl}. \\
&P\_5 \rightarrow \text{cat}. \\
&P\_5 \rightarrow \text{dog}. \\
&P\_6 \rightarrow \text{young}. \\
&P\_7 \rightarrow \text{likes}. \\
&P\_7 \rightarrow \text{chases}.
\end{aligned}
$$

This grammar is isomorphic to the unknown grammar G_U.

8 Concluding Remarks

In this paper, we consider the problem of learning a context-free grammar adequate for bottom-up parsing. We make much more of the "operationality" of the grammar learned by the learning algorithm in contrast to traditional grammatical inference problems. We set up the new learning problem for context-free grammars that is slightly different from the usual grammatical inference problem. Then the grammar learned by our algorithm is quite adequate for designing bottom-up parser or efficient bottom-up parsing. Thus this problem setting makes our learning algorithm practicable.

Lastly we remark on related work. Crespi-Reghizzi [5] is most closely related, as it describes a constructive method for learning a context-free grammar from positive examples of structural descriptions. However his algorithm and our one use completely different methods and learn different classes of context-free grammars. Since our formalism is based on tree automata, one of merits of our way is the simplicity of the theoretical analysis and the easiness of understanding the algorithm, whereas the time efficiency of his algorithm [5] is still not clear. Perhaps there may be a useful synthesis of these two approaches. The investigation that we must do but still does not is the characterization of the "reversible context-free languages". Especially it is interesting to contrast them with noncounting context-free languages [6].

References

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

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

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

[4] W. S. Brainerd. The minimalization of tree automata. *Information and Control*, 13:484–491, 1968.

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

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

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

[8] Y. Sakakibara. *Inferring parsers of context-free languages from structural examples.* Research Report 81, IIAS-SIS, FUJITSU LIMITED, 1987.