数理解析研究所講究録
第 716 巻 1990 年 60-70

On Learning A Class of Context-free Languages in Polynomial Time

横森　貴
Takashi YOKOMORI

Department of Computer Science
University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182, JAPAN

Abstract. The problem of learning context-free languages is studied, in which a subclass called *c-deterministic* context-free languages is introduced. The class of c-deterministic context-free languages properly contains the class of regular sets.

It is shown that the class of c-deterministic context-free languages is learnable in polynomial time from membership queries and equivalence queries, that is, it is polynomial time learnable from so-called *minimally adequate teacher.*

1 Introduction

We consider the problem of learning a class of context-free grammars. The problem of learning a "correct" grammar for the unknown language from finite examples of the language is known as the grammatical inference problem.

The grammatical inference problem is one of the most attractive issues in many AI areas in that it may bring us fruitful implications in the field of machine learning such as syntactic pattern recognition and automatic program synthsis. Most of the existing practical methods can, however, only solve the problem for the class of regular sets and there are a few studies for more general classes(e.g.,[2], [5],[7],[9]).

Recently, Angluin gives a polynomial time algorithm for learning regular sets from equivalence queries and membership queries([2]).

In this paper, we present an algorithm for learning c-deterministic context-free languages from membership queries and equivalence queries (so-called "minimally adequate teacher") in polynomial time. Since the class of c-deterministic context-free languages properly contains the class of regular sets, this gives a generalization of the corresponding results on regular sets in [2].

2 Preliminaries

We assume the reader to be familiar with the rudiments of formal language theory(See, e.g., [4]), and we only give some basic notations and definitions used in this paper.

For a given finite alphabet Σ, the set of all strings with finite length (including *zero*) is denoted by Σ^*.(An empty string is denoted by λ.) $lg(w)$ denotes the length of a string w. Σ^+ denotes $\Sigma^* - \{\lambda\}$. A *language* L over Σ is a subset of Σ^*. For a string x in Σ^* and a languge L over Σ, let $x\backslash L = \{y|xy \in L\}(L/x = \{y|yx \in L\})$. A language $x\backslash L(L/x)$ is called *left(right) derivative of L with respect to x*. For any w in Σ^*, $Pref(w) = \{x|xy = w$ for some $y\}$ and $Suf(w) = \{x|yx = w$ for some $y\}$. Let L_1 adn L_2 be languages, then $L_1L_2 = \{xy|x \in L_1, y \in L_2\}$

A *context-free grammar* is denoted by $G = (N, \Sigma, P, S)$, where N and Σ are alphabets of *nonterminals* and *terminals* respectively such that $N \cap \Sigma = \phi$. P is a finite set of rules : each rule 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 rule of P and α and γ are any strings in $(N \cup \Sigma)^*$, then we may write $\alpha A\gamma \underset{G}{\Longrightarrow} \alpha\beta\gamma$. The notation $\overset{*}{\underset{G}{\Longrightarrow}}$ is the reflexive and transitive closure of $\underset{G}{\Longrightarrow}$. (The subscript G is abbreviated when it is clear from the context and is written as $\Rightarrow$.) For $\alpha \in N^+$, let $L(\alpha) = \{x \in \Sigma^*|\alpha \overset{*}{\underset{G}{\Longrightarrow}} x\}$. In particular, $L(S)$, denoted by $L(G)$, is called the language generated by G. A language L is *context-free* if there exists a context-free grammar G such that $L = L(G)$.

We sometimes abbreviate context-free grammars and their languages as *CFGs* and *CFLs*, respectively.

Since we are concerned with the learning problem of context-free grammars, without loss of generality, we restrict our consideration to only λ-free context-free grammars.

A context-free grammar $G = (N, \Sigma, P, S)$ is *2-standard form* if each rule is of one of the following forms: $A \rightarrow aBC$, $A \rightarrow aB$, $A \rightarrow a$, where A, B, C are nonterminals and a is a terminal symbol. A *CFG G* is *reduced* if (1) for any $A, B \in N$, $L(A) \neq L(B)$, (2) for any $A \in N$, there are derivations such that $S \Rightarrow^* \alpha A\beta$ and $A \Rightarrow^* w(w \in \Sigma^*)$.

In what follows, we may assume that any grammar G is a λ-free, reduced *CFG* in 2-standard form. Further, a derivation by the relation $\Rightarrow$ indicates the *left-most* one.

2.1 An Example

Let $G = (N, \Sigma, P, S)$ be a *CFG*. For each $A \in N$, let $A \Rightarrow a\alpha \Rightarrow^* au \in \Sigma^*$, where au is a shortest terminal string derivable from A. Then, a rule $A \rightarrow a\alpha$ is called *shortest rule* of A.

With a given *CFG G*, we can associate a finite graph C_G as follows: [step 1] For each $S \rightarrow a\alpha \in P$, connect a node S with a node α by an arrow labelled a. (S is called *starting node*. If $\alpha = \lambda$, then introduce a new special symbol called *final node* and connect S with it.) [step *i*] Let $\alpha = Au(A \in N, u \in N^*)$ be a node created at step (*i-1*). Then, apply to each α the following procedure: If A does not appear yet as the left-most nonterminal of any node created previously, then for each $A \rightarrow b\beta$, create a node βu and connect α with it by an arrow labelled b. Otherwise, connect α with the node βu, corressponding to the right-hand side of a shortest rule $A \rightarrow b\beta$, by an arrow labelled b. (If βu is λ, then connect $\alpha(= A)$ with the final node.) Let i be $i + 1$ and repeat step i until the procedure cannot be applicable to any node.

It is easily seen that the procedure terminates in finte time. The graph C_G is called the *characteristic graph of* G([10]). The graph C_G contains the complete information on the grammar G, and more importantly, each nonterminal in N is characterized by a finite set of paths in C_G. Let us see it below.

Take the following CFG as an example for our discussion: $G = (\{S, A, B, C, D\}, \{a, b, c\}, P, S)$, where P is given by

$$
\begin{array}{l}
S \rightarrow aAD|aD, \quad A \rightarrow aAB|aB \\
D \rightarrow bCA, \quad B \rightarrow b, \quad C \rightarrow c.
\end{array}
$$

Note that the language $L(G)$ is $\{a^m b^m c a^n b^n | m, n \geq 1\}$. The characteristic graph of G is pictured in Figure 1.

By a simple calculation, we have the following equations:

$$
\begin{array}{rlll}
a\backslash L(S) = & L(A)L(D) \cup L(D) & \cdots & (1) \\
ab\backslash L(S) = & L(C)L(A) & \cdots & (2) \\
abc\backslash L(S) = & L(A) & \cdots & (3) \\
aaab\backslash L(S) = & L(B)L(D) & \cdots & (4) \\
L(D)/bcab = & \{\lambda\} & \cdots & (5)
\end{array}
$$

Let $L = L(S)$. Then, from (3)

$$L(A) = abc\backslash L$$

is immediately obtained. Further, from (4) and (5), it holds that

$$L(B) = aaab\backslash L/bcab.$$

In the same manner,

$$
\begin{array}{l}
L(C) = ab\backslash L/ab \\
L(D) = aab\backslash L
\end{array}
$$

are obtained.

Thus, given an $L(= L(G))$ each nonterminal X of G is completely characterized by a pair (x, z) such that $x \in Pref(w), z \in Suf(w)$, for some $w \in L$. That is, each nonterminal of G has its own context (x, z) by which it is distinguished from others. This feature will play a significant role in the process of learning a class of $CFGs$, and leads us to the following presented in the next section.

Notes.

(1) Let y_X be a shortest terminal string derivable from X, then, for each pair (x, z) such that $L(X) = x\backslash L/z$, a string xy_Xz is always in L and is corresponding to a path from S to the final node in C_G.

(2) For each $X \in N$, a pair (x, z) such that $L(X) = x\backslash L/z$ is not necessarily unique. For

example, $L(A) = a\backslash L/bcab$(module λ).

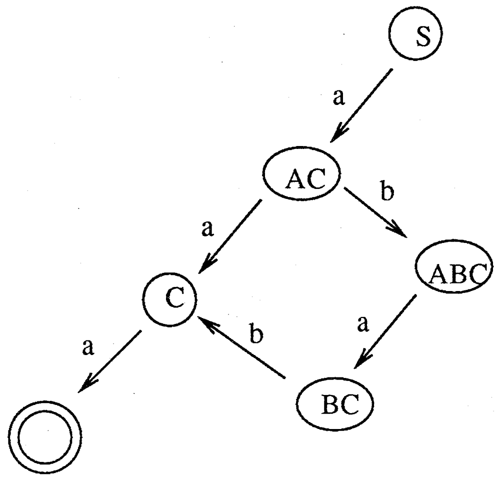

Figure 1. Characteristic Graph C_G

2.2 C-Deterministic CFGs

Let $G = (N, \Sigma, P, S)$ be a CFG. A nonterminal A in N is *context-deterministic*(abb.,*c-deterministic*) iff there is a pair (x, z) such that $S \Rightarrow^* xAz$ $(x \in \Sigma^+, z \in \Sigma^*)$ and $L(A) = x\backslash L(G)/z$. A CFG G is *c-deterministic* iff each nonterminal A in N is c-deterministic. A language L is *c-deterministic* iff there exists a c-deterministic CFG G such that $L = L(G)$.

For $A \in N$, let d_A be a derivation :$S \Rightarrow^* xA\alpha \Rightarrow^* xy_A\alpha \Rightarrow^* xy_Az_\alpha(x, y_A \in \Sigma^+, z_\alpha \in \Sigma^*, \alpha \in N^*)$ with the properties that (1) there is no duplicated application of an identical rule in the derivation of x, (2)y_A is a shortest string derivable from A, and (3) z_α is a shortest string derivable from α.

Let Q_A be the set of triples (x, y_A, z_α) of all d_As.(It is easy to see that Q_A is finite.)

Lemma 1 *Let $G = (N, \Sigma, P, S)$ be a c-deterministic CFG. Then, for any A in N, there exists (x, y_A, z) in Q_A such that $L(A) = x\backslash L(G)/z$.*

Proof. Let $L = L(G)$. Since G is c-deterministic, for each $A \in N$, there exists a pair (x, z) such that $L(A) = x\backslash L/z$ and $S \Rightarrow^* xAz$. Let y_A be a shortest string in $L(A)$.

Suppose that (x, y_A, z) is not in Q_A, and that

$$S \Rightarrow^* x'Az' \Rightarrow x'\beta z' \Rightarrow^* x'uAvz' = xAz, \text{where} \quad u, v, x', z' \in \Sigma^*, \beta \in (N \cup \Sigma)^*.$$

It is clear that $L(A) \subseteq x'\backslash L/z'$ holds, from which we have $u\backslash L(A)/v \subseteq x'u\backslash L/vz'(= x\backslash L/z = L(A))$. It is obvious that $L(A) \subseteq u\backslash L(A)/v$. Hence, $L(A) = u\backslash L(A)/v$ holds. This implies that for u, v above, there is no other derivation such that $A \Rightarrow^* u\alpha v$, for some $\alpha(\neq A) \in N^+$. Hence, if there is an application of the rule $A \rightarrow \beta$ during the derivation process from S to $x'Az'$, one may replace $x'Az'$ with $x''Az''$ (such that $S \Rightarrow^* x''Az'' \Rightarrow^* x''uAvz''$ and $lg(x'z') > lg(x''z'')$) so that $x''u\backslash L/vz'' = L(A)$ holds.

In this manner, after applying the above procedure repeatedly, we eventually obtain a pair (x', z') such that, besides $L(A) = x'\backslash L/z'$, a triple (x', y_A, z') satisfies the requirements of d_A. □

Further, as shown below, the set Q_A is obtained from the characteristic graph of the grammar at issue.

Lemma 2 *Let C_G be the characteristic graph of a CFG $G = (N, \Sigma, P, S)$. Then, without counting self-looping, the length of a path in C_G is less than $2|P|^2 + |P|$, where $|P|$ is the cardinarity of P.*

Proof. From the definition of C_G and the property of a grammar in 2-standard form, the length of the longest path is less than $|N|(2t + 1)$, where $|N|$ is the cardinality of N, $t = \max_{A \in N} \{y_A | y_A$ is a shortest string in L(A) $\}$. Since $|N| \leq |P|$ and $t \leq |P|$, the longest path is bounded by $2|P|^2 + |P|$ in length. □

Let $t_G = 2|P|^2 + |P|$ and $R_G = \{w \in L(G) | lg(w) \leq t_G\}$. Furhter, let w be in Σ^* such that $lg(w) \geq 2$. Then, $Non(w)$ is defined as $\{(x, y, z) | x, y \in \Sigma^+, z \in \Sigma^*$ and $xyz = w\}$. Finally, let $NT(G) = \bigcup_{w \in R_G} Non(w)$.

Lemma 3 *For any A in N of a c-deterministic CFG G, there is a triple (x, y, z) in $NT(G)$ such that $L(A) = x\backslash L(G)/z$.*

Proof. From the way of constructing C_G, each triple (x, y, z) in Q_A just corresponds to a string associated with each path in C_G. Hence, Q_A is a subset of $NT(G)$. □

This guarantees that the set $NT(G)$, depending only on the size t_G and $L(G)$, can provide complete information on all nonterminals of G, which implies that nonterminal membership queries are replaceable in terms of membership queries to an appropriate subset of $NT(G)$, as discussed below.

3 Learning CFGs

3.1 Learning Protocols

Let L be a target CFL over a fixed alphabet Σ. We assume the following types of queries in the learning process. A *membership query* proposes a string $x \in \Sigma^*$ and asks whether $x \in L$ or not. The answer is either *yes* or *no*.

An *equivalence query* proposes a grammar G and asks whether $L = L(G)$ or not. The answer is *yes* or *no*, and in the latter case together with a counterexample w in the symmetric difference of L and $L(G)$. A counterexample w is *positive* if it is in $L - L(G)$, and *negative* otherwise.

The learning protocol consisting of membership queries and equivalence queries is called *minimally adequate teacher.*

The purpose of the learning is to find a CFG $G = (N, \Sigma, P, S)$ such that $L = L(G)$ with the help of minimally adequate teacher.

In [1], Angluin employs a strong query called *nonterminal membership queries* which, given a string $x \in \Sigma^*$ and a nonterminal A of G with unknown set of rules P, can ask whether $x \in L(A)$ or not, and the answer is *yes* or *no.* The primary issue here is how one can replace nonterminal membership queries by membership queries at the sacrifice of some sort of restriction.

3.2 Diagnosing Rules

Let $G = (N, \Sigma, P, S)$ be a CFG, where $N = \{A_1(= S), ..., A_n\}$. A *replacement* σ is a finite tuple $[(y_1, B_1), ..., (y_t, B_t)]$, where $y_i \in \Sigma^*$, $B_i \in N$. For $\beta \in (N \cup \Sigma)^*$, σ is *compatible with* β iff there exist $x_0, ..., x_t \in \Sigma^*$ such that $\beta = x_0 B_1 x_1 B_2 \cdots B_t x_t$. Suppose σ is compatible with β. Then, an *instance* of β by σ, denoted by $\sigma[\beta]$, is a terminal string obtained from β by replacing each occurrence of B_i with a terminal string y_i.

A rule $A \rightarrow \alpha$ (not neccessarily from P of G) is *incorrect for* $L(G)$ iff there exists a replacement $\sigma = [(y_1, B_1), ..., (y_t, B_t)]$ which is compatible with α such that, for each $i = 1, ..., t$, $y_i \in L(B_i)$ and $\sigma[\alpha] \notin L(A)$. A rule is *correct for* $L(G)$ iff it is not incorrect for $L(G)$. Note that each rule of P is correct for $L(G)$.

The *diagnosis procedure* is essentially the same as Angluin's one([1]) and a special case of Shapiro's one([6]). The input is a correct parse tree $T_{A,w}$ for a *conjectured* grammar G such that A is the label of the root, w is the yield string of the tree not in L. (That is, w is a negative counterexample to L. The output is a rule that is incorrect for L.

The diagnosis procedure considers in turn each child of the root of $T_{A,w}$. If the child is labelled with nonterminal B and has a yield x, then the procedure tests if x is in $L(B)$ or not. If the answer is *no*, then it calls itself recursively with the sub-parse tree rooted at the child. If the answer is *yes*, then it goes on the next child of the $T_{A,w}$. If all the queries are answered *yes*, then the diagnosis procedure returns the rule $A \rightarrow \alpha$ at the top of $T_{A,w}$, which is incorrect for L.

3.3 Producing Candidate Rules

Let $G = (N, \Sigma, P, S)$ be a (conjectured) grammar obtained in the learning process. Whenever a new positive counterexample w is given, the set of nonterminals N is updated as follows: $N := N \cup Non(w)$. Further, construct $P_{new} = \{A \to a\alpha | a \in \Sigma, lg(\alpha) \leq 2, A\alpha \in N^+$, and $A\alpha$ contains at least one new element of N $\}$. Then, let $P := P \cup P_{new}$.

Note that through Section 3 the similar kind of argument can be found in the context of learning simple deterministic languages in [3].

3.4 Learning Algorithm

[Algorithm A]

Input : a c-deterministic CFL L over Σ.
Output : a CFG G in 2-standard normal form such that $L = L(G)$;

Procedure :
set $G = (\{S\}, \Sigma, P, S))$, where $P = \phi$;
repeat
make an equivalence query to G ;
If the answer is a *positive* counterexample w, **then**
introduce new nonterminals from w and add them to N ;
add all candidate rules to P;
else if the anwer is a *negative* counterexample w, **then**
diagnose P of G ;
remove an incorrect rule from P ;
until the answer of the equivalence query is *yes*
output G and halt.

The correctness of the algorithm A is based on the principle so-called *contradiction backtracing algorithm* originally discussed in [6]. Papers [1] and [3] apply this principle in the context of derivation process of a context-free grammar.

Important facts are 1) whenever a positive counterexample is given, at least one triple corresponding to a new nonterminal (not in the conjectured grammar) is introduced, and 2) whenever a negative counterexample is given, at least one rule incorrect for $L(G_0)$ of a correct grammar G_0 is removed from the conjectured grammar.

Theorem 4 *Given a c-deterministic context-free language L over a fixed Σ, the algorithm A halts and outputs a context-free grammar G such that $L = L(G)$.*

3.5 Time Efficiency

Suppose $G_0 = (N_0, \Sigma, P_0, S)$ is a c-deterministic CFG such that $L = L(G_0)$. The *size of* G_0 is defined by $|N_0| + |P_0|$.

Lemma 5 *For a given positive counterexample w, the number of elements of $Non(w)$ is at most $\frac{1}{2}lg(w)(lg(w)-1)$, and hence, is computable in time polynomial in $lg(w)$.*

Proof. Easy and omitted. □

Lemma 6 *The number of required positive counterexamples is at most $|N_0|$.*

Proof. From the property of the algorithm A, it is not until new nonterminals are necessary to derive a string in the target language L that a positive counterexample is given. Further, whenever a positive counterexample is given, the conjectured grammar gains at least one new nonterminal (x, y, z) corresponding to a nonterminal A in N_0, which is assured by Lemma 3. Thus, the number of required positive counterexamples is not greater than $|N_0|$. □

Lemma 7 *The number of triples introduced as nonterminals by the algorithm A is bounded by $\frac{1}{2}|N_0|m_p(m_p-1)$, where m_p is the maximum length of positive counterexamples.*

Proof. From Lemma 5, each time a positive counterexample w is given, at most $\frac{1}{2}lg(w)(lg(w)-1)$ number of nonterminals is introduced. Hence, from Lemma 6, the total number of nonterminals (triples) introduced in the entire process of leanrnig is bounded by $\frac{1}{2}|N_0|m_p(m_p-1)$, where m_p is the maximum length of positive. □

Theorem 8 *The running time of the algorithm A is bounded by a polynomial in the size of G_0 and the maximum length of counterexamples.*

Proof. The algorithm relies on three subrocedures,.i.e., the computation of candidate rules, parsing, and diagnosis.
(a) Comutation of candidate rules: Note that a conjectured grammar G is asummed to be in 2-standard form. From Lemma 7, the total number $r_L(|N_0|, m_p)$ of rules constructed in the entire process of learning L is bounded by :

$$p(|N_0|, m_p) \times |\Sigma| \times (p(|N_0|, m_p)+1)^2, \text{where} \quad p(x, y) = \frac{1}{2}xy(y-1).$$

(b) Parsing: It is well known that there exists an algorithm which , given a CFG G and a string w in $L(G)$, produces a parse tree $T_{S,w}$ in time proportional to $|G|lg(w)^3$(e,g, [4]). Since $|G| = |N| + |P| \leq p(N_0, m_p) + r_L(|N_0|, m_p)$, each parsing requires at most $(p(N_0, m_p) + r_L(|N_0|, m_p))m_n^3$.
(c) Diagnosis: Given a parse tree $T_{S,w}$, there are at most $lg(w)$ nonterminals appearing in it. Hence, the diagnosis procedure makes at most $lg(w)(\leq m_n)$ membership queries in order to find a rule incorrect for $L(G_0)$.

Now, there are at most $|N_0|$ times when positive counterexamples are provided. Each time a negative counterexample is provided, one incorrect rule is removed from P of a conjectured grammar G. This implies that the number of negative counterexamples required is not greater than $r_L(|N_0|, m_p)$.

Thus, the total time the algorithm A requires is bounded by:

$$N_0 \times r_L(|N_0|, m_p) + r_L(|N_0|, m_p) \times \{(p(N_0, m_p) + r_L(|N_0|, m_p))m_n^3 + m_n\}$$
$$\leq |G_0| r_L(|G_0|, m_c) + r_L(|G_0|, m_c)\{(p(|G_0|, m_c) + r_L(|G_0|, m_c))m_c^3 + m_c\}$$

where $m_c = Max\{m_p, m_n\}$ is the maximum length of counterexamples. □

3.6 Implications

Let $G = (N, \Sigma, P, S)$ be a CFG. A nonterminal A in N is *harmonic* iff $S \Rightarrow^* uAv$ and $S \Rightarrow^* u'Av'$ imply $u\backslash L(G)/v = u'\backslash L(G)/v'$. A CFG G is *harmonic* iff so is each nonterminal A in N. A language L is *harmonic* iff there exists a harmonic CFG G such that $L = L(G)$([8]).

Lemma 9 *A harmonic linear CFL is c-deterministic.*

Proof. Let L be a language such that $L = L(G)$ for some harmonic linear CFG $G = (N, \Sigma, P, S)$. Without loss of generality, we may assume that for each $A(\neq S) \in N$, every rule with the nonterminal A in the left-side is of one of the forms:

$$A \rightarrow \alpha A\beta, A \rightarrow \alpha' B\beta', A \rightarrow w$$

where $A, B \in N$, $\alpha, \alpha', \beta, \beta', w \in \Sigma^*$, and B derives no string containing A. Hence, for each $A \in N$, $L(A)$ is infinite.

We claim that G can be modified to have the property that $\forall x, y \in \Sigma^*$, $S \Rightarrow^* xAy$ iff $S \Rightarrow^* xBy$ imply that $L(A) = L(B)$.

Suppose otherwise, i.e., $\forall x, y \in \Sigma^*$, $S \Rightarrow^* xAy$ iff $S \Rightarrow^* xBy$ and that $L(A) \neq L(B)$. (Note that both $L(A)$ and $L(B)$ are infinite.) It is easy to see that since $\forall x, y \in \Sigma^*$, $S \Rightarrow^* xAy$ iff $S \Rightarrow^* xBy$, each recursive rule $A \rightarrow \alpha A\beta$ must be exactly corresponding to $B \rightarrow \alpha' B\beta'$, i.e., $\alpha = \alpha'$ and $\beta = \beta'$. For the same reason, it must hold a non-recursive rule $A \rightarrow \alpha B\beta$ is in P iff $B \rightarrow \alpha A\beta$ is in P. By introducing new nonterminal $[A, B]$, merge each two corresponding rules into one new rule $[A, B] \rightarrow \alpha[A, B]\beta$ and remove old rules. Further, for all of other non-recursive rules $A \rightarrow \gamma$ ($B \rightarrow \gamma'$), remove them and add $[A, B] \rightarrow \gamma|\gamma'$. Then, with $[A, B]$ replace all occurrence of A and B in the right-side of rules. After applying the above procedure to all pairs of nonterminals inconsistent to the claim, the resulting grammar clearly satisfies the requiremet of the claim. This construction preserves the equivalence of the resulting grammar to the original one, including the harmonicity.

Hence, we have an equivalent harmonic linear grammar with the property that $\forall u, v \in \Sigma^*[\ S \Rightarrow^* uAv$ iff $S \Rightarrow^* uBv]$ implies that $L(A) = L(B)$. Thus, for $\forall x, z \in \Sigma^*$, if $S \Rightarrow^* xAz$, then $x\backslash L/z = L(A)$. □

Corollary 10 *The class of harmonic linear CFLs is polynomial time learnable from minimally adequate teacher.*

Note that since the class of harmonic *linear* $CFLs$ properly contains the class of regular sets, the class of c-deterministic $CFLs$ properly contains the class of regular sets.

A CFG $G = (N, \Sigma, P, S)$ is *simple deterministic*(abb.,SD) iff $A \to a\alpha$ and $A \to a\beta$ are in P imply that $\alpha = \beta$([4]). A nonterminal A in N is *suffix-free* iff x is in $L(A)$ implies that for all $y \in Suf(x) - \{x\}$ y is not in $L(A)$. A CFG G is *suffix-free* iff so is each nonterminal A in N.

Lemma 11 *A suffix-free SDG* $G = (N, \Sigma, P, S)$ *is c-deterministic.*

Proof. From the property of SDG's, it holds that if $S \Rightarrow^* xA\alpha$, then $L(A\alpha) = x\backslash L$. Since G is suffix-free, $L(\alpha)$ is suffix-free. Hence, let w be a string in $L(\alpha)$, then $L(A) = x\backslash L/w$ is obtained. □

Corollary 12 *The class of suffix-free SDLs is polynomial time learnable from minimally adequate teacher.*

4 Discussions

We have presented an algorithm for learning a class of context-free languages from minimally adequate teacher, which is based on the characterization results of nonterminals using derivatives of a target language. The class of c-deterministic $CFGs$ has been targeted, and it was shown the algorithm learns a correct grammar in polynomial time from minimally adequate teacher, which gives a generalization of the corresponding result on regular sets in [2].

However, it should be mentioned that the definition of a minimally adequate teacher in this paper is slightly different from the original one by Angluin ([2]) in that the latter assumes the class of conjectures to be the same type as target class(i.e., both classes comprise finite-state automata and their regular sets, respectively), while the former allows arbitrary $CFGs$ in 2-standard form as conjectures in order to learn a subclass of $CFLs$ called c-deterministic. This kind of teacher is called *extended* minimally adequate teacher in [3].

In [8] harmonic linear $CFGs$ are introduced and a complete learning algorithm from positve and negative examples is given without time analysis, which is clearly shown to be not less than NP-complete from the problem setting. Note that the c-deterministic language in Section 2.1 is not harmonic linear.

The paper [3] gives a polynomial time algorithm for simple deterministic $CFLs$ from the same learning protocol as the one in this paper. When the class of $SDLs$ is targeted, the algorithm in this paper works as a simpler version of the one in [3].

The class of simple deterministic $CFLs$ is incomparable to the class of c-deterministic or harmonic linear $CFLs$ discussed here, and all of them properly contain the class of regular sets.

References

[1] D. Angluin. *Learning k-bounded context-free grammars.* Res.Rep. 557, Dept. of Comput. Sci., YALE Univ., 1987.

[2] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[3] H. Ishizaka. *Learning simple deterministic languages.* Tech. Rep. 465, ICOT, Tokyo, 1989. Also in Proc. COLT'89.

[4] M.A.Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, MA, 1978.

[5] Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. In *Proceedings of 1st Workshop on Computational Learning Theory*, pages 296–310, 1988.

[6] E. Shapiro. *Inductive inference of theories from facts.* Res.Rep. 192, Dept. of Comput. Sci., YALE Univ., 1981.

[7] Y. Takada. Grammatical inference for even linear languages based on control sets. *Information Processing Letters*, 28:193–200, 1988.

[8] K. Tanatsugu. A grammatical inference of harmonic linear languages. *International Journal of Computer and Information Sciences*, 13:413–423, 1984.

[9] T. Yokomori. Inductive inference of contex-free languages based on context-free expressions. *International Journal of Computer Mathematics*, 24:115–140, 1988.

[10] T. Yokomori. *Learning simple languages in polynomial time.* Tech. Rep., SIG-FAI, Japanese Society for AI, June, 1988.