Language Learning with Characteristic Examples and Membership Queries

Sakamoto, Hiroshi
Research Institute of Fundamental Information Science Kyushu University

RIFIS-TR-CS-108

RIFIS Technical Report

Language Learning with Characteristic Examples and Membership Queries

Hiroshi Sakamoto

April 4, 1995

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812-81, Japan
E-mail:hiroshi@rifis.kyushu-u.ac.jp Phone:092-641-1101 Ext.4459

Language Learning with Characteristic Examples and Membership Queries

Hiroshi Sakamoto
Research Institute of Fundamental Information Science
Kyushu University 33, Fukuoka 812-81, Japan
E-mail: hiroshi@rifis.kyushu-u.ac.jp

坂本 比呂志
〒812-81 福岡市東区箱崎 6-10-1
九州大学理学部附属基礎情報学研究施設

Abstract

This paper introduces the notion of characteristic examples for languages and shows that the notion contributes to language learning in polynomial time. A characteristic example of a language L is a string of L which includes, in a sense, sufficient information to represent the language L. We show that any context-free language can be divided into a finite set of languages each of which has a characteristic example. We prove that it is solvable whether or not a context-free language has a characteristic example. Then, we propose a learning model with membership queries and characteristic examples for the class of parenthesis languages. We prove that, for this class, our learning algorithm runs in a polynomial time in the size of a minimal parenthesis grammar which generates the target language and in the length of a longest characteristic example given to the algorithm.

1 Introduction

An algorithm which learns a language with queries receives yes/no answers according to criteria of the queries. Angluin [Ang87a] introduced a learning model with two types of queries, a membership query and an equivalence query. For a membership query of a string, the oracle returns 'yes' if the string is an element of the target language; otherwise it returns 'no'. For an equivalence query of a language, the oracle returns 'yes' if the language is equivalent to the target language; otherwise it returns a counterexample.

After her work, many researchers [Sak90, Ang87b, BR87, Tak88] have developed language learning with queries. Most of their methods are based on the notion of observation tables introduced by Angluin [Ang87b]. Ishizaka [Ish90] proposed another method based on the theory of model inference [Sha83] and showed that the class of simple deterministic languages can be learned in a polynomial time.

Sakakibara [Sak90] showed that the class of context-free grammars can be learned by a model using membership queries, equivalence queries and structural strings. However, there is a big open problem whether or not the class can be learned by more restricted model such as a model using only membership queries and equivalence queries. In this paper, we restrict ability of oracles and propose a new model for learning formal languages, which assumes two types of oracles. One is an oracle to answer membership queries of a target language, and the other is an oracle to select examples from a target language.

In general, examples are arbitrarily given to a learning algorithm. Hence, some of them do not contribute to learning of the target language. In fact, in case a target language is divided into some disjoint sub-languages, and no example is given from one of such sub-languages, the algorithm can never identify the whole language. If we can always select elements which contribute to language learning, an algorithm will efficiently learn a target language using such elements. Hence, in this paper, we assume our oracle with this ability, that is, the oracle divides a target language into a finite number of sub-languages each of which has a kind of representative elements of the sub-language, and gives such elements to the algorithm as examples. Such sub-languages and representative elements are said to be *complete* languages and *characteristic examples*, respectively.

We apply our learning model to the class of parenthesis languages which was introduced by McNaughton [McN67], and show that the class can be learned by our algorithm in a polynomial time. In Section 3, we define characteristic examples for formal languages. We prove that it is solvable whether or not a context-free grammar has a characteristic example. In Section 4, we present a learning algorithm for parenthesis languages with membership queries and characteristic examples. In Section 5, we show the correctness of our algorithm for parenthesis languages. Furthermore, we show that our algorithm runs in a polynomial time in the size of a minimal parenthesis grammar and in the length of a longest characteristic example.

2 Preliminaries

An alphabet is a finite non-empty set of distinct symbols. For an alphabet Σ, Σ^* denotes the set of all finite string of symbols from Σ. For a finite set S, $|S|$ denotes the cardinality of S. A language L over Σ is a subset of Σ^*.

A *context-free grammar* is a 4-tuple $G = (N, \Sigma, P, S)$, where N and Σ are alphabets such that $N \cap \Sigma = \phi$, $S \in N$, P is a finite set of rules of the form $A \to w$ $(A \in N, w \in (N \cup \Sigma)^*)$. N, Σ, P and S are said to be a set of nonterminals, a set of terminals, a set of rules, and a start symbol, respectively.

For strings $\alpha, \beta \in (N \cup \Sigma)^*$, a binary relation $\Rightarrow$ is defined as follows: $\alpha \Rightarrow \beta$ if and only if there exist strings $\gamma_1, \gamma_2 \in (N \cup \Sigma)^*$ and a rule $A \to w \in P$ such that $\alpha = \gamma_1 A \gamma_2$ and $\beta = \gamma_1 w \gamma_2$. The relation $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$. $L(G)$ denotes the set of terminal strings derived from the start symbol of the grammar G.

Let r be a rule of a grammar. The size $||r||$ of r denotes the length of right side of r. Let P be the set of rules of a grammar. The size $||P||$ of P denotes $\Sigma_{r \in P} ||r||$. Let $G = (N, \Sigma, P, S)$ be a grammar. The size $||G||$ of G denotes $|N| + |\Sigma| + ||P||$.

A derivation tree of $G = (N, \Sigma, P, S)$ is a tree such that each internal node is labeled by an element of N, each terminal node is labeled by an element of Σ and, for each internal node labeled by $A \in N$, there exists a rule $A \to w(\in P)$, where $w \in (N \cup \Sigma)^*$ is the concatenation of labels of its children in left-to-right order.

A grammar G is said to be *unambiguous* if, for any terminal string $w \in L(G)$, there exists exactly one derivation of w from G. A grammar is said to be *backwards-deterministic* if no two rules of the grammar have the same right side. A string β is said to be a context if β has one blank symbol. For a context β, $\beta[w]$ denotes the string replaced its blank by a string w. Two nonterminals A_1 and A_2 of a grammar G are said to be *equivalent* if, for any context β, either both $\beta[A_1]$ and $\beta[A_2]$ are derived from G or neither are. A nonterminal A of a grammar G is useless if there exists no context β such that $\beta[A]$ is derived from G or no terminal string is derived from A. A grammar is said to be *reduced*, if no two distinct nonterminals of it are equivalent and it has no useless nonterminal.

A *parenthesis grammar* is a context-free grammar all of whose rules are of the from $A \to (w)$, where w contains no occurrence of (or of). We note that the following properties hold for parenthesis grammars [McN67].

1. Every parenthesis grammar is unambiguous.

2. For any parenthesis grammar G, there exists a reduced backwards-deterministic parenthesis grammar G' such that $L(G) = L(G')$.

In this paper, by a parenthesis grammar we mean a reduced backwards-deterministic parenthesis grammar.

3 Characteristic Example

3.1 Definition and proposition

Definition 1 A terminal string w is said to be a *characteristic example* of a grammar G if there exists a derivation tree of G for w in which all rules of G are used. A grammar G is said to be *complete* if G has a *characteristic example.*

There exists a grammar G which is not complete. In fact, for any grammars $G_1 = (N_1, \Sigma_1, P_1, S)$ and $G_2 = (N_2, \Sigma_2, P_2, S)$, let $G = (N, \Sigma, P, S)$ a grammar with $N = N_1 \cup N_2'$, $\Sigma = \Sigma_1 \cup \Sigma_2$ and $P = P_1 \cup P_2'$, where N_2' is obtained from N_2 as follows: if $A \in N_1 \cap N_2$, then replace A of N_2 by A' ($A' \notin N_1 \cup N_2$), and P_2' is obtained from P_2 by this replacement. Clearly, the grammar G is not complete.

For grammars $G_1, G_2, \cdots, G_i$ ($G_j = (N_j, \Sigma_j, P_j, S), 1 \leq j \leq i$), let $\cup_{j \leq i} G_j$ denote a grammar $G' = (N', \Sigma', P', S)$ such that $N' = \cup_{j \leq i} N_j$, $\Sigma' = \cup_{j \leq i} \Sigma_j$ and $P' = \cup_{j \leq i} P_j$.

For grammars $G_1 = (N_1, \Sigma_1, P_1, S)$ and $G_2 = (N_2, \Sigma_2, P_2, S)$, $G_1 \sqsubseteq G_2$ denotes that $N_1 \subseteq N_2$, $\Sigma_1 \subseteq \Sigma_2$ and $P_1 \subseteq P_2$.

Proposition 1 Let G be a context-free grammar and n be the number of rules of G. There exists a set $S_G = \{G_1, G_2, \cdots, G_k\}$ of complete grammars such that $L(S_G) = L(\cup_{i \leq k} G_i) = L(G)$ and $k \leq n$.

Proof. Let n be the number of rules of a context-free grammar $G = (N, \Sigma, P, S)$. There exists a set S_G of complete grammars such that $L(S_G) = L(G)$ and $|S_G| \leq 2^n - 1$.

Remove a set $\{G_{i_1}, G_{i_2}, \cdots, G_{i_j}\}$ from S_G if there exists a grammar $G' \in S_G$ such that $\cup_{k \leq j} G_{i_k} \sqsubseteq G'$, and remove a grammar G' from S_G if there exists a set $\{G_{i_1}, G_{i_2}, \cdots, G_{i_j}\} \subseteq S_G$ such that $G' \sqsubseteq \cup_{k \leq j} G_{i_k}$. Then, for any grammar $G_i \in S_G$, there exists a rule r of G_i such that no grammar $G_j \in S_G - \{G_i\}$ has r. Thus, $|S_G| \leq n$. $\square$

For a grammar G, a set S_G of grammars which satisfies the condition in Proposition 1 is said to be a set of *complete* grammars *with respect to* G.

Proposition 2 Let G be a parenthesis grammar and $S = \{G_1, G_2, \cdots, G_m\}$ be a set of complete grammars with respect to G. Let W_i and W_j be sets of characteristic examples from G_i and G_j ($1 \leq i, j \leq m$), respectively. W_i and W_j are disjoint if and only if G_i is not equal to G_j.

Proof. If $G_i = G_j$, then $W_i = W_j$. Let $G_i \neq G_j$. Then, there exists a rule r of G_i such that r is not a rule of G_j. Each $w_i \in W_i$ is derived using r at least once. Since G is unambiguous, W_i and W_j are disjoint. $\square$

3.2 Decision problem

Definition 2 Let G be a grammar and A be a nonterminal of G. An occurrence of A is said to be *bounded* if there exists a constant k such that, in any derivation tree of G, A occurs at most k.

Definition 3 Let G be a grammar and r be a rule of G. An occurrence of r is *bounded* if there exists a constant k such that, in any derivation tree of G, r is used at most k.

For nonterminals A, B and C of a grammar, $A \Rightarrow_C^* B$ denotes that C occurs in a derivation from A to B. For nonterminals A, B and a rule r of a grammar, $A \Rightarrow_r^* B$ denotes that r is used in a derivation from A to B.

For a derivation tree d, let $|d|$ denote the depth of d. For a set N of nonterminals, let d_N denote a derivation tree in which every $A \in N$ occurs. For a set P of rules, let d_P denote a derivation tree in which every $r \in P$ is used.

Lemma 1 Let G be a context-free grammar and A be a nonterminal of G. It is solvable whether or not the occurrence of A is bounded.

Proof. The occurrence of $A \in N$ is not bounded if and only if there exists $B \in N$ such that $B \Rightarrow_A^* B$. For a context free grammar $G = (N, \Sigma, P, S)$ and any $\alpha, \beta \in (N \cup \Sigma)^+$, it is solvable whether or not $\alpha \Rightarrow^* \beta$. Hence, for $A \in N$, it is solvable whether or not there exists $B \in N$ such that $B \Rightarrow_A^* B$. $\square$

Lemma 2 Let G be a context-free grammar and r be a rule of G. It is solvable whether or not the occurrence of r is bounded.

Proof. Let $G = (N, \Sigma, P, S)$ be a context-free grammar. The occurrence of a rule r of G is not bounded if and only if there exists $B \in N$ such that $B \Rightarrow_r^* B$.

If r is of the form $A \to w$ ($A \in N, w \in \Sigma^+$), there exists $B \in N$ such that $B \Rightarrow_r^* B$ if and only if there exists $B \in N$ such that $B \Rightarrow_A^* B$.

If r is of the form $A \to \alpha C \beta$ ($C \in N, \alpha, \beta \in (N \cup \Sigma)^*$), there exists $B \in N$ such that $B \Rightarrow_r^* B$ if and only if there exists $B \in N$ such that $B \Rightarrow_A^* B$ and $B \Rightarrow_C^* B$. Thus, from Lemma 1, Lemma 2 holds. $\square$

Lemma 3 Let $G = (N, \Sigma, P, S)$ be a context-free grammar. Let $P' \subseteq P$ be such that any $r \in P'$ is not bounded. It is solvable whether or not there exists a derivation tree of G in which every rule in P' is used.

Proof. Let $N' \subseteq N$ be such that $B \in N'$ if and only if there exists $r \in P'$ such that $B \Rightarrow_r^* B$. There exists $d_{P'}$ if and only if there exists $d_{N'}$. For any $A, B \in N$, if $A \Rightarrow^* B$, then $A \Rightarrow^k B$ ($k \leq |N|$). Then, there exists $d_{N'}$ if and only if there exists $d_{N'}$ at most depth $|N|^2$. Thus, it is solvable whether or not there exists $d_{P'}$ of G. $\square$

Theorem 1 It is solvable whether or not a context-free grammar is complete. If a context-free grammar is complete, then a characteristic example of the grammar is effectively obtained from it.

Proof. Let $G = (N, \Sigma, P, S)$ be a context-free grammar. Let $P' = \{r \in P | \ r \text{ is bounded}\}$. From Lemma 2, there exists $d_{P'}$ if and only if there exists $d_{P'}$ such that $|d_{P'}| \leq |N|$. From Lemma 3, there exists $d_{P-P'}$ if and only if there exists $d_{P-P'}$ such that $|d_{P-P'}| \leq |N|^2$. Thus, there exists d_P if and only if there exists d_P such that $|d_P| \leq |N|^2$.

Hence, we conclude Theorem 1. □

4 Learning Algorithm

First, we outline the learning algorithm for parenthesis languages. An input to our algorithm is a set of characteristic examples of the target language. For a characteristic example, our algorithm determine the derivation tree. For a derivation tree, our algorithm computes a grammar and outputs the union of such grammars as a parenthesis grammar for the target language.

To determine a derivation tree, our algorithm uses membership queries. In this section, we define a membership query for 'yield' and describe the procedure $\mathcal{M}$ to determine derivation trees for characteristic examples. The procedure $\mathcal{M}$ is a main part our algorithm.

4.1 Membership query for yield

Definition 4 A *yield* of a tree is the concatenation of all labels of leaves of the tree in left-to-right order.

Definition 5 Let k be the number of internal nodes of a tree. A *skeleton* s for the tree is uniquely obtained by replacing all labels of internal nodes of the tree by labels $\sigma_1, \sigma_2, \cdots, \sigma_k$ from root to leaves and from left to right.

Let s be a tree and a be an internal node of s. Let $s(a)$ be a subtree of s whose root is a. Let a be an internal node of a tree s and let s' be a tree. $s(a, s')$ denotes the tree obtained by replacing $s(a)$ of s by s'.

Definition 6 Let s and s' be skeletons for derivation trees of a grammar G. Let a, a' be internal nodes of s, s', respectively. The relation $\equiv_{G(s,s')}$ is defined as follows:

$$a \underset{G(s_i,s_j)}{\equiv} a' \quad \text{if and only if} \quad \text{yields of} \quad s(a, s'(a')) \quad \text{and} \quad s'(a', s(a)) \quad \text{are in} \quad L(G).$$

Definition 7 Let G be a parenthesis grammar and s be a skeleton. Then, the answer to the membership query for the yield of s is defined as follows: if the yield is in $L(G)$, then the answer *yes* is returned; otherwise the answer *no* is returned.

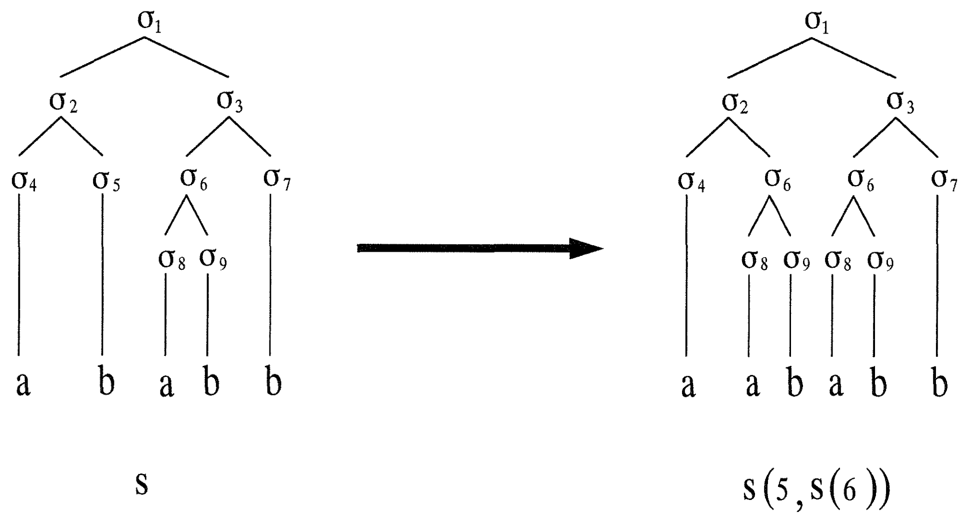

Figure 1: A skeleton and its replacement

Example 1 For a string $((((a)(b)))(((a)(b))(b)))$, Figure 1 depicts the skeleton s and its replacement. σ_i $(1 \le i \le 9)$ of s denotes a label of node i. The tree $s(5, s(6))$ is obtained by replacing $s(5)$ of s by $s(6)$. The yield of $s(5, s(6))$ is the string $(((a)((a)(b)))(((a)(b))(b)))$.

4.2 Procedure $\mathcal{M}$ for skeletons

In Figure 2, we give the procedure $\mathcal{M}$ that determines derivation trees of characteristic examples with membership queries. An input is a set of skeletons for characteristic examples of complete grammars with respect to a target parenthesis grammar. An output is a set of derivation trees for given characteristic examples.

Let $\hat{s} = \{s_1, s_2, \cdots, s_m\}$ be a set of skeletons for m characteristic examples of a target language L_U. First, for any two nodes i and j of a skeleton s_k $(1 \le k \le m)$, $\mathcal{M}$ uses a membership query. For such nodes i and j of a skeleton s_k, a membership query proposes two yields of trees $s_k(i, s_k(j))$ and $s_k(j, s_k(i))$. If these two yields are both in L_U, then the answer yes is returned; otherwise the answer no is returned. If the answer yes is returned, then $\mathcal{M}$ renames σ_j by σ_i if $i < j$ or renames σ_i by σ_j if $j < i$.

Then, for any node i' of a skeleton s_i and any node j' of other skeleton s_j $(1 \le i \le m-1, i < j)$, $\mathcal{M}$ uses a membership query. For such nodes i' and j', a membership query proposes the two yields of trees $s_i(i', s_j(j'))$ and $s_j(j', s_i(i'))$. If the two yields are both in L_U, then the answer yes is returned; otherwise the answer no is returned. If the answer yes is returned, then $\mathcal{M}$ renames $\sigma_{j'}$ of s_j by $\sigma_{i'}$ of s_i if $i' < j'$ or renames $\sigma_{i'}$ of s_i by $\sigma_{j'}$ of s_j if $j' < i'$. If the answer no is returned and $i' = j'$, then $\mathcal{M}$ renames $\sigma_{j'}$ of s_j by $\sigma_{k'}$, where no skeleton $s_k \in \hat{s}$ has a label $\sigma_{k'}$.

Finally, $\mathcal{M}$ outputs a refined $\hat{s}$ as the set of derivation trees of a grammar for the target

```
**Procedure** $\mathcal{M}$
**Input:** a set $\hat{s} = \{s_1, s_2, \cdots, s_m\}$ of skeletons for $m$ characteristic examples
**Output:** derivation trees for given characteristic examples
**begin**
  **foreach** $s_i \in \hat{s}$ $(1 \leq i \leq m)$                                /* First loop */
    **foreach** nodes $j, k$ of $s_i$ **do**
      make membership queries;
      if $j \equiv_{G_U(s_i,s_i)} k$, then
        rename $\sigma_k$ by $\sigma_j$ $(j < k)$ or $\sigma_j$ by $\sigma_k$ $(k < j)$;
      else;
  **foreach** $s_i, s_j \in \hat{s}$ $(1 \leq i \leq m-1, i < j)$                     /* Second loop */
    **foreach** nodes $i'$ of $s_i$ and $j'$ of $s_j$ **do**
      make membership queries;
      if $i' \equiv_{G_U(s_i,s_j)} j'$, then
        rename $\sigma_{j'}$ by $\sigma_{i'}$ $(i' < j')$ or rename $\sigma_{i'}$ by $\sigma_{j'}$ $(j' < i')$;
      if $i' \not\equiv_{G_U(s_i,s_j)} j'$ and $i' = j'$, then
        rename $\sigma_{j'}$ by a new label $\sigma_{k'}$
      else;
  output $\hat{s}$;
**end**
```

Table 1: The procedure $\mathcal{M}$ to decide derivation trees

parenthesis language L_U. Our algorithm computes a parenthesis grammar $G = (N, \Sigma, P, S)$ using such a refined $\hat{s} = \{s_1, s_2, \cdots, s_m\}$. Since each s_i $(1 \leq i \leq m)$ is a skeleton for a characteristic example, each symbol in Σ occurs in a skeleton. Hence, Σ is computable. For any two labels of internal nodes of skeletons, it is decidable whether or not they are equal. Hence, N is also computable. For a skeleton, if there exists an internal node whose label is σ such that it has children whose labels are $\sigma_1, \sigma_2, \cdots, \sigma_m$ $(m \geq 1)$ in left-to-right order, then our algorithm makes a rule $\sigma \to (\sigma_1\sigma_2 \cdots \sigma_m)$ of G.

In the next section, we prove that, for any parenthesis language, our algorithm outputs a correct parenthesis grammar.

5 Correctness and Complexity

Lemma 4 Let a and b be internal nodes of a derivation tree d of a parenthesis grammar G. Two labels of a and b are equal if and only if $a \equiv_{G(d,d)} b$.

Proof. Clearly, if two labels of a and b are equal, then $a \equiv_{G(d,d)} b$. We assume $a \equiv_{G(d,d)} b$. Let σ_a, σ_b denote labels of a, b, respectively. Since G is backwards-deterministic, if two yields of $d(a)$ and $d(b)$ are equal, then σ_a and σ_b are equal.

Let two yields of $d(a)$ and $d(b)$ are not equal. G has two rules of the form $\sigma \to (w_1\sigma_a w_2)$ and $\sigma' \to (w_1'\sigma_b w_2')$ for nonterminals σ and σ', strings w_1, w_2, w_1' and w_2'. G also has two rules of the form $\sigma \to (w_1 \sigma_b w_2)$ and $\sigma' \to (w'_1 \sigma_a w'_2)$. Then, any $\beta[\sigma_a]$ and $\beta[\sigma_b]$ derived from G are of the form $W_1(w_1\sigma'' w_2)W_2$ or $W'_1(w'_1\sigma'' w'_2)W'_2$ for a nonterminal $\sigma'' \in \{\sigma_a, \sigma_b\}$, strings W_1, W_2, W'_1 and W'_2. Thus, $\beta[\sigma_a]$ is derived from G if and only if $\beta[\sigma_b]$ is derived from G. Since G is reduced, no two distinct nonterminals of G are equivalent. Hence, σ_a and σ_b are equal. □

Lemma 5 Let a, a' be internal nodes of derivation trees d, d' of a parenthesis grammar G, respectively. Two labels of a and a' are equal if and only if $a \equiv_{G(d,d')} a'$.

Proof. Since G is unambiguous, for any $w \in L(G)$, there exists exactly one derivation tree of G. Then, similarly to Lemma 4, we can prove Lemma 5. □

From Lemma 4 and Lemma 5, we conclude that, for a target parenthesis language, our algorithm eventually terminates and outputs a parenthesis grammar which generates the target parenthesis language.

We now analyze the time complexity for our algorithm.

Lemma 6 The time required for $\mathcal{M}$ is bounded by a polynomial in the number of characteristic examples and in the length of a longest characteristic example.

Proof. It is sufficient to show that the number of membership queries is bounded by a polynomial. Let k be the number of internal nodes of a skeleton for a characteristic example w. The number of membership queries for w is at most $\frac{1}{2}k(k-1)$.

Let m be the number of characteristic examples and n be the number of internal nodes of a skeleton for a longest characteristic example. The number of membership queries in the first loop and in the second loop of $\mathcal{M}$ are at most $\frac{1}{2}mn(n-1)$ and at most $\frac{1}{2}m(m-1)n^2$, respectively. Thus, the number of membership queries is in $\mathcal{O}(m^2n^2)$. □

Lemma 7 The number of characteristic examples given to $\mathcal{M}$ is bounded by a polynomial in the size of a minimal parenthesis grammar.

Proof. Let $G = (N, \Sigma, P, S)$ be a parenthesis grammar. From Proposition 1, the number of characteristic examples to learn $L(G)$ can be bounded by $||P||$ $(< ||G||)$. □

From these lemmas, we have the following theorem.

Theorem 2 There exists an algorithm that learns a parenthesis language with membership queries and characteristic examples, and runs in a polynomial time in the size of a minimal grammar for the target parenthesis language and in the length of a longest characteristic example given to the algorithm.

6 Concluding Remarks

We have introduced the notion of characteristic examples for languages. We have discussed properties of characteristic examples and proved that it is solvable whether or not a context-free grammar has a characteristic example.

We have presented an algorithm that learns parenthesis languages. We also have shown that the time required for our algorithm is bounded by a polynomial in the size of a minimal parenthesis grammar and in the length of a longest characteristic example.

One open problem we have not answered is whether or not any other interesting sub-classes of contex-free grammars can be learned by our learning model. Since every context-free grammar has an equivalent reduced backwards-deterministic grammar and the decision problem in Section 3 is solvable, it seems that the definition of characteristic examples can be applied to such sub-classes. However, in general, context-free grammars are ambiguous. Hence, for a string derived from a context-free grammar, there may exist two or more derivation trees. In order to overcome this difficulty, we need a new method to determine derivation trees.

References

[Ang88] Angluin, D. *Queries and concept learning.* Machine Learning, 2:319-342, 1988.

[Ang87a] Angluin, D. *Learning k-bounded context-free grammars.* Technical Report YALEU/DCS/RR-557, Department of Computer Science, Yale University, 1987.

[Ang87b] Angluin, D. *Learning regular set from queries and counter-examples.* Information and Computation, 45:117-135, 1987.

[BR87] Berman, P. & Roos, R. *Learning one-counter language in polynomial time.* In Proceedings of 28th IEEE Symposium on Foundations of Computer Science, pages 61-67. IEEE Computer Society Press, 1987.

[Har78] Harrison, M.A. *Introduction to Formal Language Theory.* Addison-Wesley, 1978.

[Ish90] Ishizaka, H. *Polynomial time learnability of simple deterministic languages.* Machine Learning, 5:151-164, 1990.

[McN67] McNaughton, R. *Parenthesis Grammars.* Journal of the ACM, 14:490-500, 1967.

[Sak90] Sakakibara, Y. *Learning context-free grammars from structural data in polynomial time.* Theoretical Computer Science, 76:223-242, 1990.

[Sha83] Shapiro, E.Y. *Algorithmic program debugging.* Cambridge, MA: MIT Press, 1983.

[Tak88] Takada, Y. *Grammatical inference for even linear languages based on control sets.* Information Processing Letters, 28:193-199, 1988.